VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2010 Lukáš Meca

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

RIA technologie, popis a příklady použití
RIA Technology, Description and Examples

2010 Lukáš Meca

Prohlášení

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. Května 2010 .....

Rád bych na tomto místě poděkoval svému vedoucímu, Ing. Radoslavu Fasugovi, Ph.D, za jeho vedení, cenné rady a celkovou odbornou pomoc, bez níž by tato práce nevznikla.

Abstrakt

Obsahem této bakalářské práce je ukázka vhodné demonstrace problematiky budování Rich Internet Application a seznámení s dostupnými nástroji pro tvorbu těchto technologií. Jelikož online rezervace sportovišť je součástí praktické části, bylo nutné popsat problematiku takového systému společně s jeho finančním aspektem a provést analýzu před samotným programováním. Klientská aplikace je vytvořena v programu Adobe Flex Builder 3, pro komunikaci se serverem bylo zvoleno PHP5 a jako databáze bylo zvoleno MySQL.

Klíčová slova: bakalářská práce, RIA, Rich Internet Application, Flex, AJAX, Silverlight, Laszlo, Adobe Flex Builder, analýza, konceptuální model, ER diagram, Use Case Diagram, Data Flow Diagram, Sekvenční Diagram

Abstract

The main goal of this Bachelor thesis is suitably demonstrate problems of creating Rich Internet Application and make reader acquainted with available tools for creating these technologies. Seeing that online reservation of sports facilities is the part of the practical part it was necessary to describe problems of this reservation system with its financial aspect and make an analysis before programming. Client application is created in Adobe Flex Builder 3 platform, communication with server is provided by PHP5 and for database MySQL was chosen.

Keywords: bachelor thesis, RIA, Rich Internet Application, Flex, AJAX, Silverlight, Laszlo, Adobe Flex Builder, analysis, conceptual model, ER diagram, Use Case Diagram, Data Flow Diagram, Sequence Diagram

Seznam použitých zkratek a symbolů

RIA	- Rich Internet Application
XML	- Extensible Markup Language
AJAX	- Asynchronous JavaScript and XML
IDE	- Integrated Development Environment
DFD	- Data Flow Diagram
HTML	- HyperText Markup Language
URL	- Uniform Resource Locator
CSS	- Cascading Style Sheets
XAML	- Extensible Application Markup Language
LINQ	- Language Integrated Query
DRM	- Digital restrictions management
WPF	- Windows Presentation Foundation
WCF	- Window Communication Foundation
DLR	- Dymamic Language Runtime
BCL	- Base Class Library
CLR	- Common Language Runtime
SWF	- Small Web Format
AIR	- Adobe Integrated Runtime
WYSIWYG	- What You See Is What You Get
IDE	- Integrated Development Environment
VŠB	- Vysoká Škola Báňská
MXML	- Magic eXtensible Markup Language
AMF	- Action Message Format
RTMP	- Real-Time Messaging Protocol

Obsah

Seznam tabulek

Seznam Obrázků

1 Úvod

Internet je v životě už běžnou záležitostí. Do styku s ním přijde téměř každý, z toho důvodu, že je již součástí většiny domácností. Když jsem si vybíral téma mé bakalářské práce, nebyl jsem si zpočátku jistý, jaké si zvolit. Jelikož už mám zkušenosti s tvorbou internetových stránek, rozhodl jsem se mou práci zaměřit na tuto oblast a zdokonalit tak zároveň mé zkušenosti o technologie, se kterými jsem se buď nesetkal, nebo setkal jen okrajově. Když jsem se doslechl o takzvaných Rich Internet Aplications, začal jsem se o toto téma více zajímat. Po prokonzultování této technologie s mým vedoucím, který můj návrh schválil, jsem se jej rozhodl vybrat a vypracovat.

V textu bude čtenář průběžně seznámen s problematikou budování RIA technologií, které jsou zakomponovány do internetových stránek rezervačního systému sportovišť. Budu se rovněž zabývat problematikou finančního aspektu tohoto systému a zvolenou technologií pro jeho vytvoření. Na závěr se budu věnovat analýze samotné praktické části mé bakalářské práce a to všechno doprovázeno ilustračními obrázky.

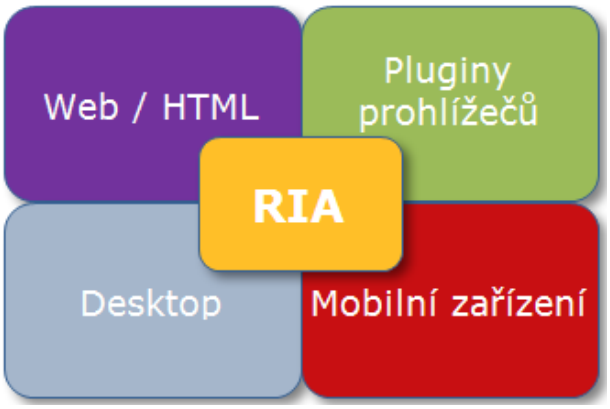
2 Seznámení s RIA technologiemi

RIA technologie se začaly vyvíjet někdy kolem devadesátých let, kdy chtěl člověk používat výhod desktopových aplikací na webu, ale nejen na něm. Zastaralé technologie se pomalu, ale jistě nahrazují RIA technologiemi. Ty nám zajistí běh aplikace na serveru, přičemž klientský počítač nám pouze renderuje stránku. To je pro uživatele výhodou, protože nemusí čekat na server při každém odkázání na nějakou stránku nebo spuštění nějaké funkce, či provedení jiné interakce. Cílem této technologie je si natáhnout všechna data a pak je jen zpracovávat. Druhou možností je získávat data postupně, na pozadí aplikace, aniž by to nějak omezovalo uživatele v dalším užívání stránky. Tato technologie překračuje hranice jazyka HTML. Dále jsou zde výhody například jako „drag and drop“ funkce, bohaté grafické možnosti a funkčnosti. Jednoduše můžeme pomocí RIA technologií vytvořit cokoliv, co si usmyslíme a co nejde vytvořit v jiných technologiích, například v HTML, css a javascriptu. Díky moderním nástrojům lze vytvořit komplexní aplikace s využitím velké škály prostředků, jako jsou různé fonty, bitmapy, vektory, animace, video, audio a další. Vznikají tak aplikace bohaté na uživatelskou potřebu a zážitek, takzvané Rich Internet Application.

Tyto aplikace jak už bylo zmíněno, nemusí běžet pouze v internetovém prohlížeči, ale je možné vytvořit plnohodnotnou desktopovou aplikaci, která má přístup k internetu. Díky tomu můžeme využívat internet bez použití prohlížeče. Podívejme se například na eBay desktop, ve kterém můžete nakupovat na internetu, aniž byste zapli prohlížeč.

Jedinou nevýhodou RIA aplikací je ta, že pro svůj chod potřebují nějaký zásuvný modul, takzvaný plugin (Java, Flash Player atd.). Tyto pluginy nepoužívá 100% uživatelů užívajících internet. Stáhnutí a nainstalování pluginu sice zabere chvilku, nicméně je to už práce navíc a přispívá k nepohodlí uživatele. Toto se ovšem mění a až se RIA aplikace dostanou širší veřejnosti, stanou se tyto pluginy běžnou potřebou uživatele. Co se může někomu zdát, jako nevýhoda je fakt, že RIA řeší čistě klientské záležitosti. Ačkoliv se existence serveru předpokládá, RIA neřeší jak je server naprogramovaný nebo na jaké platformě běží. Takže například do Adobe Flexu 3 můžeme poslat data z MySQL databáze pomocí php skriptu a to data v podobě například xml či textu. Taková data Flex dokáže jednoduše zpracovat.

Shrnutí principů, RIA technologie zahrnují oblasti znázorněné na obrázku níže:



Obrázek 1: Prvky pokryté v RIA

Hlavní rysy RIA:

- Možnost uživatelského rozhraní se velice blíží desktopovým aplikacím
- Přímá interakce
- Obnovování jen části stránky
- Offline provoz
- Snadnost spuštění RIA se blíží snadnosti spuštění webových stránek

2.1 Rozšíření RIA technologií do mobilních zařízení

RIA technologie je už nějaký čas možné spouštět na různých typech počítačů, s různými typy operačních systémů. Díky pluginům, které jsou volně dostupné, to není problém. Donedávna se to však omezovalo pouze na tyto domácí počítače či notebooky. Postupem času ovšem došlo k rychlé změně kupředu a firmy mají velkou snahu dostat RIA technologie do mobilních zařízení.

Co se týče Flashe samotného, vyšla nová verze Adobe Flash 10.1, která podporuje mnoho nových mobilních zařízení včetně systémů Google Android, Blackberry, Symbian, Palm webOS a Windows Mobile. Z toho vyplývá, že vytvořené aplikace v programu Adobe Felx builder 3 nebo 4 bude možné spustit na mobilním zařízení, které obsahuje zmiňovaný operační systém. Zde nastává ovšem problém se zařízeními typu iPod, iPhone a iPad od firmy Apple. Každá aplikace pro telefon typu iPhone musí být původně napsána v jazyce jako je Objective-C, C, C++ nebo Javascript a spouštěna skrze WebKit engine v iPhone OS. A i přes veškerou snahu firmy Adobe prosadit si u těchto zařízení integraci Flash playeru, se to zatím nepodařilo, a kdo ví, jestli k takovému kroku někdy firma Apple dá svolení.

Když jsem se porozhlédl u konkurenční firmy Microsoft, zjistil jsem, že ani zde nezůstali pozadu. Jejich Silverlight nyní běží na operačním systému jako je Symbian 3 nebo Windows Mobile 7. Brzy se také počítá s nasazením do operačního systému Android. Narozdíl od firmy Adobe dostal Microsoft svolení pro nasazení silverlightu do iPhonu. Nyní se ještě vrátím k prvním dvěma zmiňovaným operačním systémům, na kterých již silverlight běží. Plugin do prohlížeče poběží na verzi Symbian^1. Verze Symbian^3, která je opensource, není podporována. Silverlight běží na limitovaném počtu telefonů jako jsou například Nokia 5800 XpressMusic, N97 a N97 mini a funguje pouze u základního prohlížeče zabudovaného v mobilním zařízení. Co se týče Windows Mobile, zde funguje silverlight 3 s některými funkcemi ze silverlight 4. Microsoft ale plánuje udělat Silverlight jako základní platformu pro Windows Mobile 7.

3 Zástupci RIA technologií

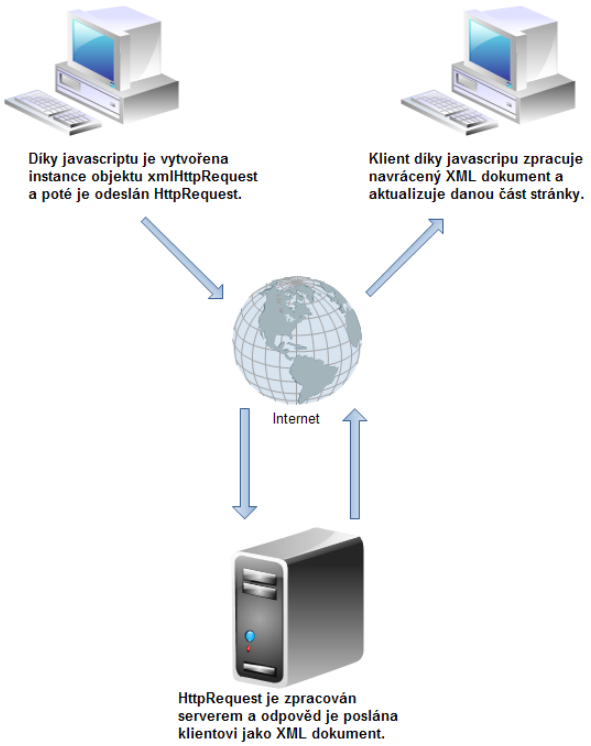
Mezi zástupce RIA technologií můžeme vyjmenovat tyto nejzákladnější:

• AJAX – ačkoli někteří AJAX nepovažují za čistou RIA technologii, z důvodu různých nevýhod, je dobré ho zde zmínit, protože určité základní rysy RIA má.
• Microsoft Silverlight
• Adobe Flash a Flex
• Laszlo
• JavaFX

3.1 AJAX

Asynchronous JavaScript and XML neboli AJAX, který je dnes často používán, využívá zavedených technologií, které jsou dnes velmi rozšířené. AJAX je jedna z prvních technologií, která pomohla ke vzniku RIA a jeho největší výhodou je, že je podporován už téměř všemi prohlížeči. Tudíž není nutnost instalace dodatečného pluginu. S AJAXem má uživatel pocit mnohem větší plynulosti práce na internetových stránkách, která se (zejména u rychlejšího připojení) blíží běžným desktopovým aplikacím. Jelikož při použití AJAXu není potřeba znova a znova načítat celou stránku, ale pouze její část, kterou potřebujeme aktualizovat, vzniká tímto i menší zátěž na server. Žádáme jen o část dat a nemusíme sestavovat celý HTML dokument.

Navzdory rozšířenosti AJAXu se zde ale vyskytují jisté nevýhody a problémy. Za největší problém AJAXu se dá s jistotou označit to, že je znemožněno použití tlačítka zpět respektive vpřed. Uživatelé jsou na tlačítko zpět respektive vpřed zvyklí a očekávají od něj, že bude fungovat. Při použití těchto tlačítek může dojít k různým situacím. V lepším případě tlačítko neudělá nic. Může taky dojít k tomu, že po jeho použití bude chování naprosto neočekávané a pravděpodobně zruší uživateli práci, kterou do té doby vykonal. Při změnách stránky za použití AJAXu se nemění naše URL adresa v prohlížeči, a proto není možno uložit tento stav stránky do záložky nebo poslat e-mailem a podobně. Tento problém se dá vyřešit za pomocí identifikátorů v URL adrese, které začínají znakem #. Při zobrazení stránky s takovou URL adresou javascript zjistí hodnoty těchto identifikátorů a uvede stránku do příslušného stavu. Problém ovšem nastává, když má uživatel v prohlížeči vypnuto využívání javascriptu. Další nevýhodou je, že existuje spousta prohlížečů. Vyladit aplikaci tak, aby vypadala a běžela na všech prohlížečích stejně, zabere kus práce a samozřejmě od toho se i odvíjí náklady. Dále technologie CSS a HTML jsou omezené, jelikož byly kdysi vytvořeny primárně pro zobrazování textu. Další nevýhodou AJAXU je, že javascript je docela výkonnostně slabší a má mnoho odlišností oproti tradičním jazykům jako je například C# java nebo ActionScript.



Obrázek 2: AJAX - princip fungování

I když se může zdát, že nevýhod je u AJAXu více než výhod, neznamená to, že by byla tato technologie špatná. Právě naopak, AJAX je velice kvalitní technologií, ale vývojář webových aplikací musí vědět, kdy ho použít.

3.1.1 Doporučená literatura

Díky obrovské rozšířenosti AJAXu, můžeme najít nezměrné množství textů, byť už anglicky nebo česky psaných. Stejně tak je spousta videotutoriálů a knížek, které se dají sehnat na internetu nebo v knihkupectví. Z česky psaných knih bych doporučil knihu od autorů Cristian Darie, Bogdan Brinzarea, Filip Chereches Tosa, Mihai Bucica s názvem AJAX a PHP tvoříme interaktivní webové aplikace [1]. Z anglicky psaných bych doporučil knihu od autora Anthony T. Holdener s názvem Ajax: The Definitive Guide [2], jelikož je novějšího vydání a obsahově velice kvalitní.

3.2 Silverlight

Silverlight byl vytvořen společností Microsoft a pro zobrazení používá plugin (lightweight plug-in). Princip Silverlightu je v podstatě stejný jako Flex a to takový, že uživatelské rozhraní je definováno ve značkovacím jazyku XAML. O výkonnou část se pak stará objektově orientovaný jazyk a díky tomu, že se v silverlightu používá .NET, získává tímto výhodu použitelnosti několika jazyků jako jsou: C#, Visual Basic, ale také Ruby, Python nebo PHP. Další

výhodou oproti Flexu je LINQ (Language Integrated Query), který umožňuje mnohem snazší dotazování se na databázi. Dále, co se týče dostupnosti vývojářů, je mnohem více těch, kteří programují v .NETu, než Flex vývojářů.

Nejnovější veze je nyní Silverlight 4. Silverlight dokáže využít akcelerace grafických karet. To znamená, že může bez problému použít na streamované video běžné filtry. Další vymožeností je možnost spouštění aplikace jako desktopovou aplikaci, bez nutnosti použití prohlížeče. V silverlightu se vyskytovaly problémy, jako jsou použití tlačítka zpět, vpřed nebo ukládání aktuální stránky do záložek. Tento problém se v třetí verzi Silverlightu vyřešil.

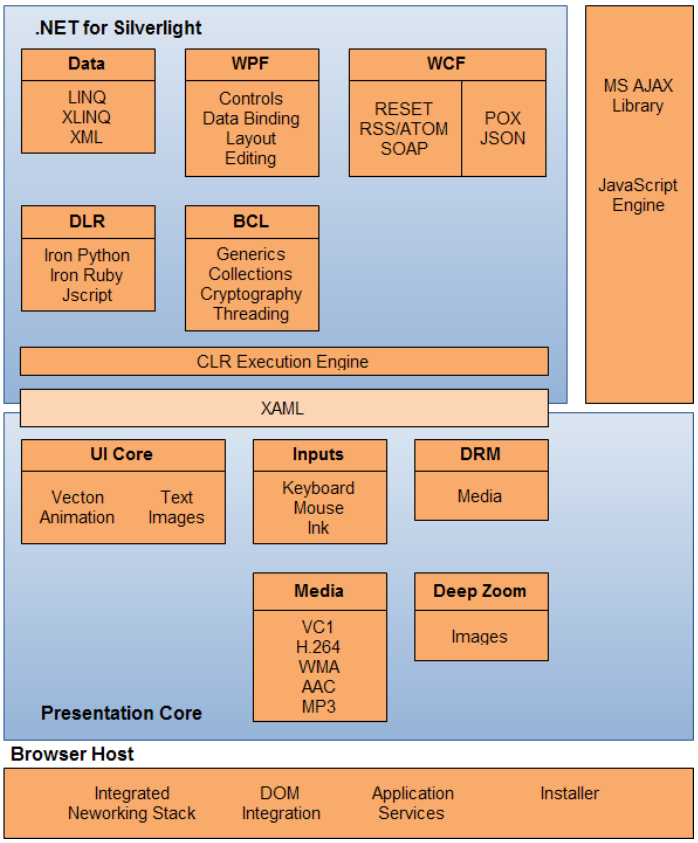
V následující tabulce je znázorněn vybraný přehled změn v jednotlivých verzích Silverlightu. Jelikož verze 1.0 byla první, nejedná se o novinky, ale jeho původní charakteristické znaky:

verze Silverlightu	vylepšení, novinky, charakteristické znaky
1.0	podpora VC-1 kodeku, WMV videa, MP3 a WMA audia, podpora 2D grafiky a animací, aplikační logika je zde pouze pomocí JavaScriptu, podpora Ajaxu, cross-platform a cross-browser plugin pro RIA aplikace
2.0	podpora Visual Basic.NET, C#, IronPython, IronRuby, Multi-Threading, nové komponenty(TextBox, RadioButton, Slider, Calendar, DatePicker, DataGrid, ListBox, TabControl, and others), Visual State Manager, LINQ, podpora XML, JSON, RSS, Data Binding, podpora nahrávání souborů
3.0	možnost spouštění na desktopu, podpora 3D, podpora grafických karet, textové animace, bitmap API, bitmapové a komponentové kešování, binární XML
4.0	podpora použití lokálních fontů, podpora tisknutí, podpora webkamer a mikrofonů, oficiální podpora pro google chrome, Bi-directional text, nastavení oken mimo prohlížeč(velikost, pozice, atd.), Managed Extensibility Framework - jedná se o sdílení rozhraní mezi jednotlivými částmi programu

Tabulka 1: Verze Silverlightu a jejich novinky

U aplikací, které jsou vytvořeny v silverlightu platí podobné podmínky, které jsou u běžných webových stránek. Například je možné v takové aplikaci vytvářet soubory, či k nim přistupovat. Tyto soubory ovšem musí být uloženy na speciálním místě, takzvaném izolovaném uložišti. Toto uložiště funguje na podobném principu jako cookies v obyčejné internetové stránce. Tyto soubory jsou zpravidla odděleny podle uživatele a aktuálních stránek a mají také omezenou velikost.

Na následujícím obrázku architektury jsou zobrazeny moduly prezentačního jádra *UI Core*, *Inputs*, *Media*, *Deep Zoom* a *DRM (Digital restrictions management)*, dále pak moduly .NET Framework platformy pro Silverlight *Data*, *WPF (Windows Presentation Foundation) WCF (Window Communication Foundation)*, *DLR (Dymamic Language Runtime)*, *BCL (Base Class Library)* a *CLR (Common Language Runtime)*.



Obrázek 3: Architektura Silverlight

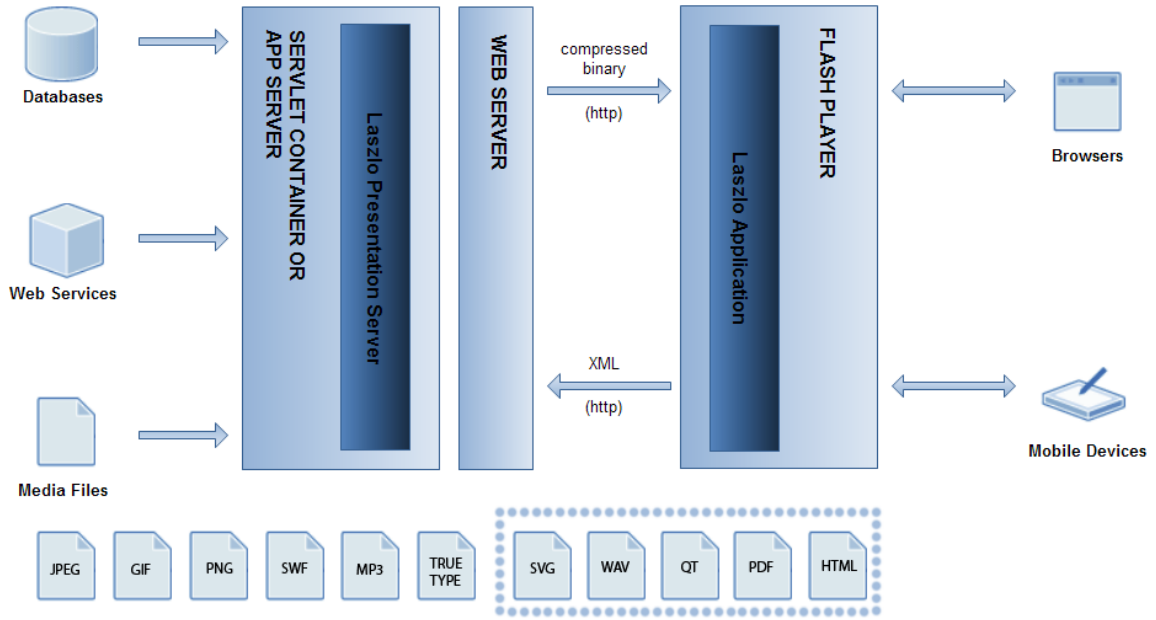
Silverlight je docela mladý a není veřejnosti tak znám jako je to u Flashe a Flash playeru, který vlastní přes 90% lidí používajících internet. Toto se mi zdá být jako jedna z největších nevýhod silverlightu a není jisté, jestli se dostane lidem do širšího povědomí, jako se to stalo u Flashe. Podle textu na stránkách [3] existuje v současné době podle čísel Microsoftu více jak 350 miliónů instalací silverlightu, přišlo s ním do styku okolo 300 tisíc vývojářů a designerů. Existuje přes 10 tisíc webů používajících silverlight. Například zimní olympijské hry v roce 2010 zprostředkovávaly přenos několika kanálů olympijských her, které byly zprostředkovávány prostřednictvím silverlightu. Na Flash s jeho více jak 90% rozšířeností to ale zdaleka nestačí.

3.2.1 Doporučená literatura

Co se týče doporučené literatury v češtině, tak bych doporučil knihu od autora John Papa s názvem Silverlight - datové služby [4], která se zabývá v prvé řadě přenosem a zpracováním dat. Dále pak vysvětluje podrobný přístupu k datům prostřednictvím Silverlightu. Ten kdo má datové služby v Silverlightu již zvládnuté, určitě ocení knihu od autora Matthew MacDonald s názvem Pro Silverlight 3 in C# [5], napsanou v anglickém jazyce. Zde se autor zabývá detailněji animacemi, videem, zvukem, transformacemi, bitmapami a moha dalšími věcmi, což jsou ve většině případech vyžadované technologie v RIA.

3.3 Laszlo

Laszlo je založen na systému programování s již známými technologiemi jako je JavaScript a XML a následném převedení do aplikací přehrávající se ve Flash playeru.



Obrázek 4: Architektura Laszlo

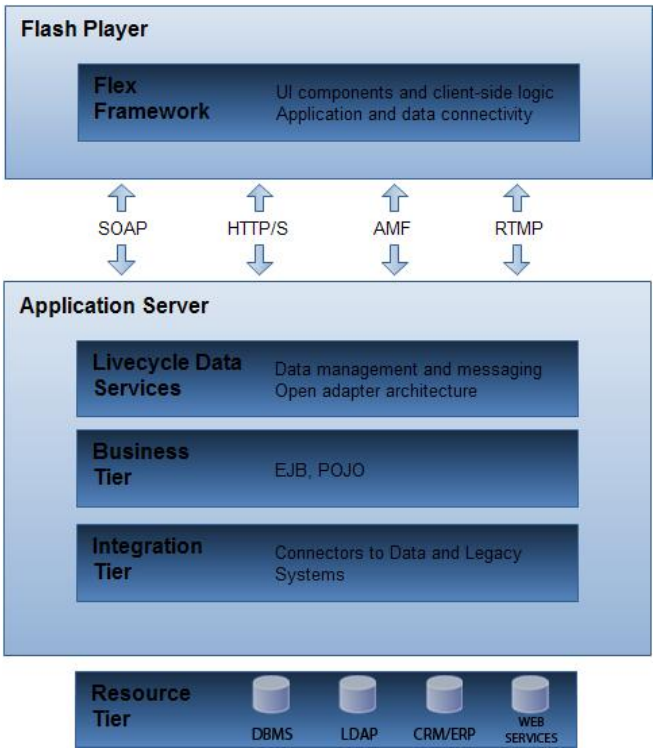
Díky převedení aplikace do flashové podoby je možné spustit aplikaci v jakémkoli prohlížeči bez ohledu na operační systém. Oproti JavaScriptu Laszlo používá kompiler, který vše převede do souboru s příponou SWF. Díky tomu se vývojáři nemusí zaobírat různými omezeními různých verzí JavaScriptu. Ke konci roku 2004 vydali Aplikaci Laszlo jako OpenLaszlo, čili open source a tím se stala přístupnou pro všechny. Používání JavaScriptu se zdálo být ze začátku jako velkou výhodou, ale v dnešní době je to právě naopak. V současné verzi totiž nemá datové typy a nepodporuje základní objektově orientované principy, jako jsou třídy a podobně.

I když se stalo Laszlo open source, stále je velmi málo používán uživateli z důvodu rozšíření mnohem silnějších vývojových prostředí.

3.4 Flex 3

Flex byl vytvořen firmou Adobe na podobném principu jako Laszlo a to takovém, že se vytvoří aplikace jako textový soubor v daném jazyce. Následně je konvertována do souboru, který je přehráván Flash playerem. Jak už bylo zmíněno, je to velice dobrý tah z důvodu rozšíření Flash playeru, který má přes 90% uživatelů internetu. Díky tomu má Flex velmi slibnou budoucnost. Oproti Flashi, který byl s trochou nadsázky využíván jen pro online video, sem tam nějakou tu hru

a otravné reklamy, se svět dočkal velmi zajímavé platformy Flex. Tato platforma má velmi dobrý a kvalitní komponentový framework, výborný objektově orientovaný programovací jazyk ActionScript 3, přívětivé IDE (které jsem si velmi oblíbil), a přitom můžeme říct, že běží téměř na všech počítačích světa.



Obrázek 5: Architektura Flex

Z pohledu architektury je Flex podobný AJAXU. Dokáže aktualizovat uživatelské rozhraní, přijímat, zpracovávat a odesílat data na pozadí aplikace. Flex je primárně vytvořen k tvorbě aplikací, nikoli animací nebo banerů. Obsahuje sadu předdefinovaných komponent a tříd, které můžeme použít. Samozřejmě si můžeme vytvářet komponenty své a přizpůsobit je našemu účelu. Dále jsou zde dvě běhová prostředí. Pro běh Flex aplikací v prohlížeči se používá Adobe Flash player. Pro instalaci a běh aplikace na desktopu se používá Adobe AIR (Adobe Integrated Runtime), díky kterému jsme mohli být svědky nástupu mnoha kvalitních aplikací, které by bez tohoto prostředí a Flexu vznikaly jen stěží.

Jak už bylo naznačeno, Flex není Flash, jak se mnoha lidem může zprvu zdát. Nevytváří se zde animace, banery a další grafické animace. Nepoužívá se zde časová osa, jako je tomu u Flashe. Naopak je založen na používání komponent a využívání plnohodnotného objektově orientovaného jazyka ActionScript 3. Flash aplikace se ukládá v binárním souboru .fla a k jeho editaci je zapotřebí Flash IDE. Zatímco zdrojové kódy vytvořené ve Flexu jsou prosté textové soubory, které můžeme editovat v jednoduchém textovém editoru. Toto je velmi příjemná změna oproti Flashi.

S daty pracujeme převážně ve formátu XML. Aplikace ve Flexu je klientská záležitost, tudíž se vůbec nezajímá o server. Pokud tedy chceme dostat data z databáze, musí se k tomu použít například PHP skript, který načte data z databáze a ve formátu XML je pošle Flexu. Co se

týče dat, má Flex velice dobrou vestavěnou podporu jejich validace. Dokáže validovat jak čísla či slova, tak například i emaily atd.

Dále, co se začínajícím Flex programátorům může zdát úžasné a velice nápomocné, je velmi kvalitní WYSIWYG editor ve Flex builderu.

Když jsem s vývojem mé aplikace začínal ve Flex 3, byl Flex 4 ještě v plenkách a jeho finální verze vyšla 21. března 2010. Proto jsem zde psal převážně o výhodách Flex 3. Určitě je ale na místě zde uvést, v čem byl Flex 4 vylepšen oproti zmíněné trojce a jaké novinky se dostaly do rukou vývojářů po jeho vypuštění do světa RIA technologií. I když je Flex 3 úspěšný a je užíván širokou škálou programátorů, dá se říct, že v mnoha oblastech jeho působení dosáhlo svých mezí a bylo třeba jeho vylepšení. Vypisovat a popsat všechny novinky by bylo na celou kapitolu, a proto v následujícím seznamu popíši alespoň těchto pár základních novinek:

- Jako první příklad, který byl ve čtyřce vylepšen, je kompoziční model uživatelského rozhraní, který není ve trojce tak pružný, jak by měl být. Například využití grafických vlastností u tlačítek se omezovalo pouze na jeho popisek a případně bylo možné do něj vložit obrázek jako jeho ikonu. Pokud chtěl někdo do tlačítka vložit video nebo jiný zvolený grafický prvek, bylo nutné se uchýlit k úpravě zdrojových kódů u této komponenty, či k jejímu rozšíření nebo k takzvaného workaroundu. Toto vyřešila nová komponentová architektura „Spark“(v předchozí verzi byla komponentová architektura „MX“), která odděluje data komponenty a její chování od grafické reprezentace. Takový systém je zaveden například u komponent ve WPF či v Silverlightu.

- Hodně vylepšení bylo vytvořeno v oblasti vývojářské tvorby. Byly vylepšeny například CSS selektory, které v předchozí verzi nebyly moc propracované. Také kompilování tvořeného projektu je 2x až 7x rychlejší než tomu bylo u předchozí verze. Přepracovány byly také stavy (v angličtině takzvané „states“), které se využívají pro přechod stránky do různých stavů.

- Primitivní grafika, jako obdélníky, čáry a elipsy, je v této verzi podporována rovnou v MXML, tedy přímo ve frameworku. Uživatel ji nemusí tvořit zvlášť jako obrázky, či přímo programovat v ActionScriptu.

- Příchodem Flex 4 vznikl i nový formát FXG, který podporuje grafické prvky, o kterých je zmínka v předchozím bodu. Vznikl tak skvělý formát pro přenášení grafiky mezi různými programy od firmy Adobe, čímž se v určité míře jistě vylepší workflow mezi programátorem a designerem.

Flex 4 má obrovskou škálu vylepšení, které přispějí k rychlejšímu, kvalitnějšímu a zajisté i jednoduššímu vývoji webových či AIR aplikací. Nicméně, abych nově vypuštěnou verzi Flexu jen nevychvaloval, musím říct, že se zde objevují samozřejmě i nějaké ty nevýhody. Například nový komponentový systém „Spark“ zatím neobsahuje všechny komponenty z původní verze, takže programátor musí kombinovat oba dva komponentové systémy dohromady, což může vytvořit řadu komplikací. Když se začínající vývojář rozhodne pracovat s tímto systémem, je jisté, že se bude muset seznámit přinejmenším se dvěma komponentovými frameworky. Jednoduchý nebude zajisté ani přechod z jedné verze na druhou pro stávající Flex vývojáře.

3.4.1 Doporučená literatura

Co se týče dostupnosti výukového materiálu, existují česky psané návody. Bohužel není jich mnoho a jsou většinou v podobě internetových stránek. Pokud chce člověk větší výběr, bude se muset spokojit s anglicky psanými texty či videi. Jelikož jsem nenarazil na knihu, která by byla napsaná v češtině, můžu doporučit pouze anglicky psanou knihu od autorů Joshua Noble, Todd Anderson s názvem Flex 3 CookBook [6].

4 Problematika online rezervace

V dnešní době, kdy světu vládne internet, by si člověk řekl, že online rezervace je samozřejmostí každého sportoviště. I když je situace o něco lepší, než tomu bylo v minulosti, stále se v mnoha případech rezervace musí provádět telefonicky nebo dokonce na místě sportoviště. Podle mého názoru je poměr sportovišť s online rezervací stále mnohem nižší, než těch bez ní. Věřím tomu, že každý, kdo se rozhodl mít online rezervaci, ať už pro své sportoviště, či celý areál, svého rozhodnutí nelitoval. Ušetřil si tím spoustu práce a nespočet telefonátů. Uživatelé si řeší vše sami a jediná starost, která majiteli zbude, je už jen inkasování poplatků za využití sportoviště.

4.1 Příklady funkcí v online rezervaci

Pokud se rozhodneme implementovat online rezervaci na něčích stránkách, měli bychom si ujasnit, co bude uživatel od rezervace požadovat a jaké budou jeho nároky. Prioritou by měla být přehlednost a snadnost použití systému. Pokud je aplikace nepřehledná a složitá, uživatele odradí a v nejhorším případě přejde ke konkurenci.

Je například vhodné uživateli umožnit rezervovat jeho zvolený čas opakovaně. Uživateli je tímto ulehčeno od mnoha zbytečných kliků navíc, což zaručeně ocení. Uživatel musí mít také možnost si svou rezervaci zrušit nebo přesunout a mít informace o tom, kdy nejpozději před začátkem si může rezervaci smazat či přesunout. Je nepřípustné, aby rezervovaný čas byl hned po uložení do databáze závazný a uživatel byl tímto povinen rezervaci zaplatit.

Také pro administrátora, který se bude starat o rezervace, musí být naimplementovány přijatelné funkce, aby mohl upravit administraci na požadovaný stav. Mezi tyto funkce patří například nastavení otevírací doby sportoviště, nastavení délky jedné rezervace, blokování určitého dne, odstranění již uložené rezervace či nastavení ceny rezervace. Velmi důležitá je taky kontrola platby, které se budeme věnovat v následující kapitole.

4.2 Proč RIA

Jelikož jsem nenarazil na žádnou online rezervaci vytvořenou v technologii RIA, je podle mého názoru výhodné na tomto příkladu poukázat na přednosti, či případné nedostatky této technologie. Jednou z největších předností použitých v mé práci je grafická stránka areálu VŠB. Uživatel si díky 3D modelu může představit, kde se nachází areál sportovišť, který si vybral, kudy se k němu dostat, a na kterém sportovišti ve vybraném areálu bude sportovat. Vytvořený model areálu byl vypůjčen z mé školy a byl použit pouze v mé bakalářské práci a nebyl nikde dále šířen. Video areálu, které je prezentováno v mé aplikaci je vytvořen ve 3D Studiu MAX, kde byl tento model částečně upraven a vyrenderován do požadovaného stavu. Uživatel si tak může detailně prohlédnout, kde půjde cvičit. S tímto také souvisí výběr sportoviště. Uživatel si díky grafickému zobrazení areálu přesně vybere kurt, halu, či jiné sportoviště, takže si vytvoří nějakou vizuální představu.

Obrázek 6: Grafické zobrazení výběru sportoviště

Obrázek 7: Grafické zobrazení výběru sportoviště po kliku na sportoviště

Další z předností je, že nemusíme pokaždé renderovat celou stránku, nýbrž načítáme data a renderujeme vždy jen část stránky. Uživateli pak neustále nepřeskakuje celá stránka při přepínání na jiné odkazy. Díky tomu můžeme využít hezkých prolínajicích a jiných efektů mezi různými stavy naší stránky.

5 Finanční aspekt

Co se týče finančního aspektu internetových rezervací, měl by být navržen přívětivě jak pro uživatele, tak pro administrátora, či provozního sportoviště. Informace a stavy o platbách jsou tudíž rozděleny do dvou skupin, a to do takových, které jsou viditelné buď jen pro uživatele, a nebo ty, které vidí pouze provozní (zpravidla vidí všechny informace).

Při vybírání rezervací by měl uživatel být schopen vidět jak cenu jednotlivé rezervace, tak součet cen za všechny jednotlivé rezervace, které si přeje uložit. Návrh by měl být jednoduchý a snadno pochopitelný.

Provozní nebo administrátor by pak měl mít možnost nastavení různých cen v různých časových úsecích. Například by mělo být umožněno nastavit jinou sazbu v dopoledních a odpoledních hodinách, či v jiných časových úsecích podle uvážení zadavatele. Dále by měla být každá cena již rezervovaného času po uložení pevně daná. V opačném případě je povinnost mít aplikaci nastavenou tak, že při každé změně ceny takovéto rezervace je uživatel na tuto změnu upozorněn a musí mu být umožněno tuto rezervaci zrušit.

Programátor musí myslet také na to, že majitel systému si bude chtít zkontrolovat vybrané finance v pokladně s financemi uloženými v databázi plateb za vybraný měsíc.

5.1 Platební metody

Jakmile se uživatel rozhodne své rezervace uložit, vzniká více možností, jak provést platbu zvolených rezervací. Uživatel má možnost si po registraci zajít na recepci sportoviště, kde mu bude po předání peněžní částky připsána tato suma na jeho osobní registrovaný účet. Následně, jakmile bude chtít uživatel uložit své rezervace, bude dotyčný požádán, zda-li chce provést platbu za rezervace z jeho registrovaného účtu. Pokud ano, bude mu požadovaná částka stržena z měny vložené na účet. Nebo může zvolit druhou možnost, a to takovou, že si svou rezervaci zaplatí na místě recepce buď před začátkem nebo na konci rezervace. Provoznímu sportoviště je tímto umožněno vidět průběžně všechny rezervace. Díky tomu má kontrolu nad tím, zda je právě probíhající rezervace již zaplacena předem přes internet nebo musí dotyčný zaplatit na místě. To nám zajistí naprostý přehled o probíhajících rezervacích.

5.2 Stornování a přesunutí rezervací

Další důležitou řešenou věcí je stornování a přesouvání již uložených rezervací. Tyto dvě funkce by měly být v každém případě uživateli umožněny. Pokud se tak nestane, nastávají nám problémy, kdy musí dotyčný volat na recepci a žádat o zrušení, či přesunutí rezervace. Tím ztrácíme onu požadovanou výhodu, kdy se nemusí zvedat telefon. Samozřejmě zde musí být zavedeno nějaké opatření, kdy rezervace začne být opravdu závazná. Jinak by uživatel mohl stornovat svou rezervaci například 5 minut před jejím začátkem. Takovým způsobem jednání by znemožnil jiným sportovcům se na daný čas rezervovat místo něj a hala, či jiné sportoviště by bylo nevyužité. Proto je třeba zavést storno, či přesunutí pouze do stanoveného času před začátkem rezervace. Například uživatel si v pondělí rezervuje tělocvičnu na pátek a pokud ji nezruší nebo nepřesune den před zahájením rezervace, což je do čtvrtku, stává se rezervace závaznou. Pokud není tato rezervace zaplacena, ať už přes internetové rozhraní nebo na místě, vytvoří se u takového uživatele penále, které musí zaplatit a pokud tak neučiní, je možnost i dočasné zablokování účtu.

5.3 Penále

Jak už bylo zmíněno, do online rezervací musí být zaveden jakýsi druh penalizace. Bez něho by lidé nemuseli brát vážně své rezervace a sportoviště by tak přicházela o spoustu peněz, které by propadly kvůli nezodpovědnosti některých uživatelů.

Není penále jako penále a záleží na majiteli, jaký druh penalizace si zvolí. Z mého pohledu nejefektivnější druh penalizace je takový, když uživatel například dvakrát nepřijde a nezaplatí, účet se automaticky zablokuje. Uživatel nemůže nadále provádět rezervace, dokud se osobně nedostaví na recepci a dluh nezaplatí. Díky tomuto postupu sportoviště možná přijde o nějakou malou sumu, ale může filtrovat nezodpovědné uživatele a zamezit opakovanému neplacení. Takže v konečném výsledku přijdou o méně peněz, než kdyby tuto filtraci neprováděli.

6 Zvolená technologie pro praktickou část práce

Ve světě RIA technologií již existuje nemálo produktů, se kterými se dá dobře pracovat a vyvíjet v nich kvalitní aplikace a je na každém z nás, jaký produkt si vybere. Samozřejmě někomu vyhovuje více programování v C# nebo třeba v Javě. Někomu, kdo již pracoval s Flashem, tak zase více vyhovuje ActionScript. Jak jsem již zmínil, je to na každém z nás, ale zvolit právě tu správnou technologii nemusí být zase tak triviální.

Když jsem přemýšlel, ve kterém vývojovém prostředí budu tvořit můj rezervační systém, zaměřil jsem se na pár základních bodů. Jelikož se už několik let zabývám tvorbou internetových stránek, rozhodl jsem si rozšířit obzory u něčeho netradičního a hlavně se naučit novým technologiím. S technologií AJAX jsem již nějakou dobu pracoval. Rozhodl jsem se tudíž tuto technologii nevyužít, a protože ani Laszlo mě nijak nezaujalo, mé rozhodnutí se omezilo pouze na technologii Silverlight nebo Flex. Můj názor je, že Flex má mnohem větší potenciál, než je tomu u Silverlightu. Je to z toho důvodu, že Flash player využívá přes 90% všech uživatelů a tudíž menší pravděpodobnost, že se bude muset dodatečně instalovat. Jelikož nemám ani velké sympatie k technologii .NET, byla moje volba využití Flexu jasná.

Co se týče serverové strany aplikace, zvolil jsem skriptovací jazyk PHP, který mi zpracovává všechna data odeslaná z mé klientské aplikace a vrací data, která potřebuji zpět ve formě XML. Jako databázi jsem použil MySQL, protože tato patří mezi nejpopulárnější a zároveň je open source.

6.1 Vývojové prostředí Adobe Flex Builder 3

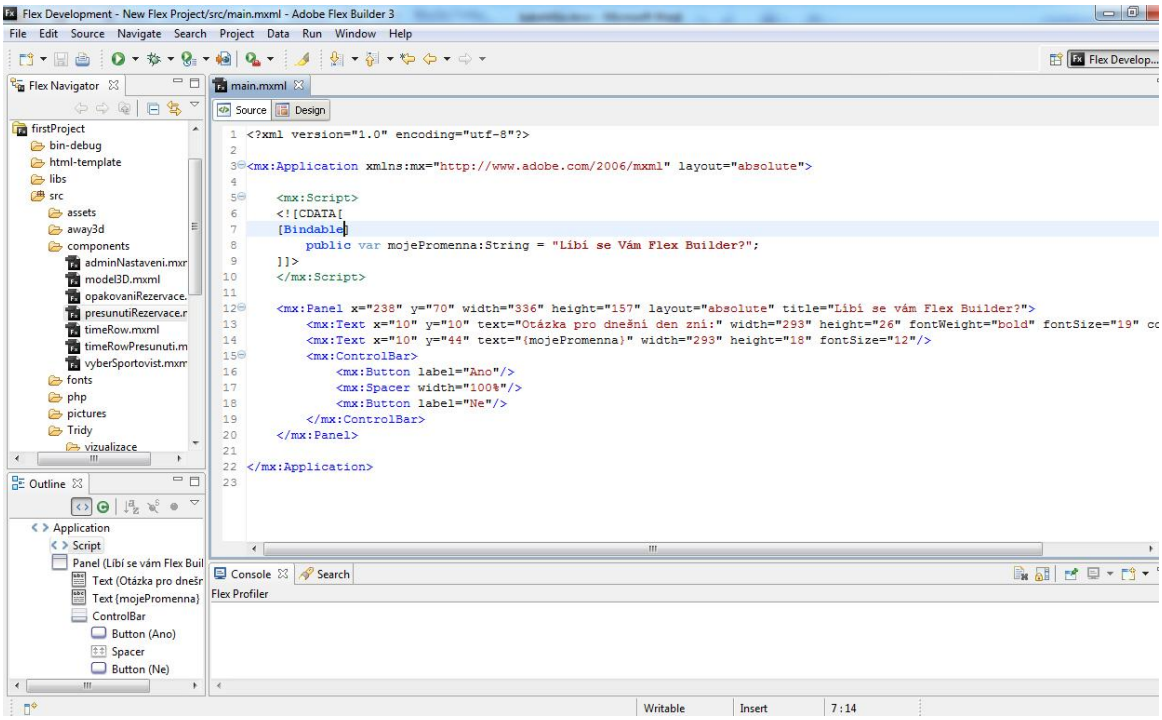
Adobe Flex Builder je vývojářský nástroj založený na softwaru Eclipse. Tento software je možné získat ve verzi Standard nebo Professional a je nabízen volně ke stažení studentům a to za účelem nekomerčního využití. V opačném případě bude zpoplatněn sumou určenou firmou Adobe. Tímto odpadly v případě tvorby mé praktické práce jakékoliv problémy s licencí. V těchto dvou verzích je umožněno importovat soubory ze softwaru Adobe Creative Suite 3, což usnadňuje práci grafickým designerům. Dále jsou ve vývojovém prostředí implementovány profilovací programy pro sledování paměti, což je velice důležitý aspekt programování.

Pokud není Flex Builder první vývojové prostředí, ve kterém programátor něco dělá, bude se mnou určitě souhlasit, že je velice jednoduché a intuitivní. Po vytvoření našeho prvního projektu můžeme vidět okno rozdělené do tří hlavních částí.

Vlevo je sloupec, kde v horní části máme stromové zobrazení našich projektů, podsložek a všech potřebných souborů. Ve spodní části levého sloupce máme kolonku nazvanou outline, kde vidíme strukturu naší aplikace a můžeme díky ní velice rychle přepínat mezi námi naimplementovanými komponentami. Ve spodní části obrazovky je pak možnost vidět všechny konzolové výpisy, varování neboli warnings při průběhu aplikace, či jiné užitečné věci.

Ve střední části obrazovky pak máme nejdůležitější část našeho vývojového prostředí a tou je okno, kde se píše samotný zdrojový kód. Tento náš zdrojový kód se dělí na dvě části. První je samotný deklarativní jazyk MXML založený na XML. Takzvané *tagy*, které píšeme do našeho kódu, jsou většinou názvy tříd, které reprezentují samotné komponenty. Například komponenta *Button*, *Label*, *Combobox*, *DataGrid* nebo novou rozšířenou datovou mřížku pro výkonnou schopnost vizualizace dat *Advanced DataGrid*. Druhou částí je samotný ActionScript kód. Je zde pro vytváření klientské logiky a díky kterému můžeme našim komponentám, ale nejen jim,

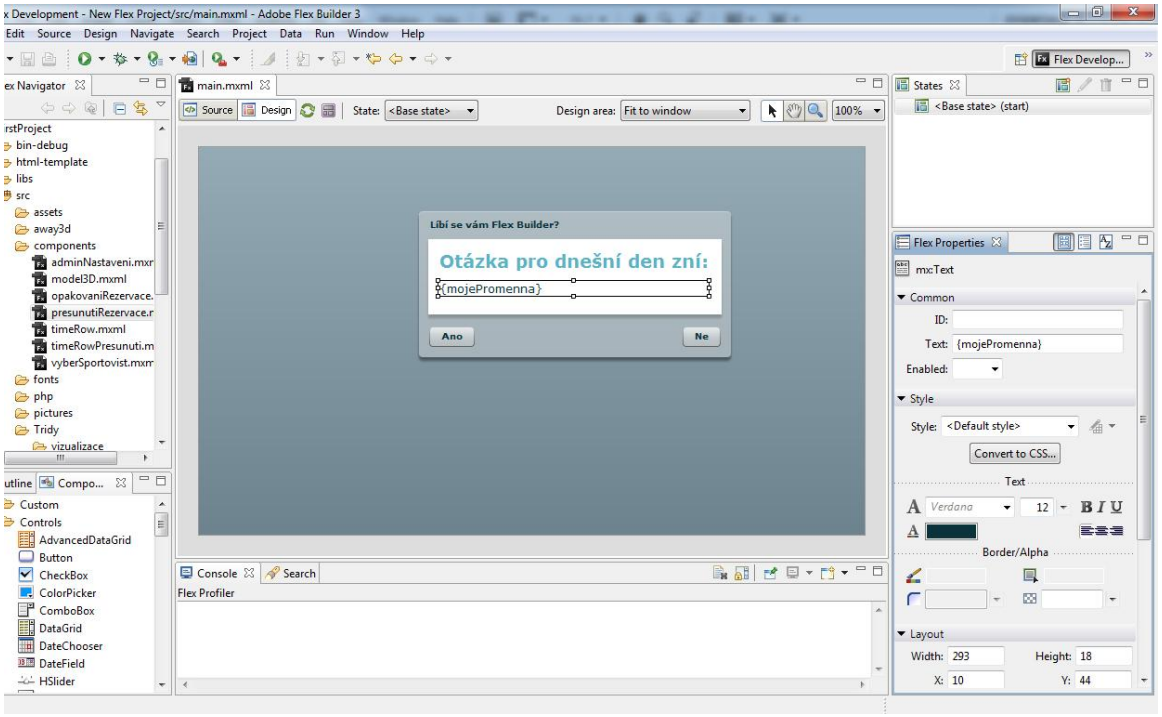
naprogramovat nějakou funkčnost. ActionScript je plnohodnotný objektově orientovaný jazyk a to znamená, že díky němu můžeme naprogramovat takřka cokoliv si budeme přát. Kód v MXML a ActionScriptu se zkompiluje dohromady do jednoho souboru SWF, který pak tvoří naši vytvořenou aplikaci Flex.



Obrázek 8: Vývojové prostředí Adobe Flex Builder 3

Flex obsahuje sestavenou knihovnu tříd a aplikačních služeb, která pomáhá vývojářům sestavovat a vytvářet aplikace RIA. Mezi tyto služby můžeme zařadit například navázání dat, řízení přetažení, zobrazovací systém, který řídí rozvržení uživatelského rozhraní, systém stylů, který řídí vzhled komponent rozhraní, efekty a systém animací, který řídí pohyb a přechody například mezi různými stavy aplikace.

Velmi silnou stránkou, kterou ocení nejen začátečníci, ale také již zkušení programátoři, je kvalitní takzvaný WYSIWYG editor. V něm si jednoduchým stylem „drag and drop“ naskládáme požadované komponenty, kde a jak si přejeme, nastavíme požadované atributy a náš Flex builder za nás vygeneruje zdrojový kód. Já osobně jsem si tento editor velmi oblíbil, protože s ním jdou velice rychle doladit určité detaily stránky.



Obrázek 9: Vývojové prostředí Adobe Flex Builder 3 - WYSIWYG

Flex Builder má také zabudovaný velmi kvalitní debugger s interaktivním krokováním, kde se můžete vnořovat do různých cyklů a detailně vidět krok za krokem, jak se váš program chová. V debuggeru lze také vidět jak všechny proměnné, tak události a jejich hodnoty, které se v našem programu objevují.

Programátoři si ve Flex Builderu mohou zvolit způsob přenosu dat, a to takový, který zahrnuje buď podporu pro XML přes komponentu HTTPService, která je využívána v mé práci, nebo formát AMF (Action Message Format) od Adobe nebo RTMP (Real-Time Messaging Protoco) s použitím softwaru Adobe LiveCycle Data Services (dříve nazývaného Flex Data Services).

I když v aplikaci vytvořené v mé bakalářské práci nepracuji s technologií AIR, ale pouze zakomponuji vytvořenou aplikaci do prohlížeče, myslím si, že stojí za to se o této technologii alespoň okrajově zmínit.

Adobe AIR je modul, který lze použít v různých operačních systémech a který vývojářům umožňuje používat známé technologie pro tvorbu internetových stránek, například HTML, Ajax, Adobe Flash nebo Adobe Flex k vytváření rozvinutých internetových aplikací určených pro stolní počítače. S runtime prostředím Adobe AIR je možné přenést námi vytvořené aplikace na plochu počítače a to tak, že budou naprosto nezávislé na prohlížeči. To nám vytváří nové možnosti, jako je například online/offline provoz s vyšším výkonem. Program Flex Builder 3 obsahuje všechny potřebné nástroje k vytváření, ladění a sbalení aplikací vytvořených na základě Adobe AIR.

Zde, stejně jako v sekci Flex 3 (viz str. 10), bych se rád zmínil o tom, že současná nejnovější verze Adobe Flex Builderu je verze 4, ale jak už jsem psal výše, když jsem s prací začínal, byla aktuální verze Flex Builder 3. Z toho důvodu byla má práce tvořena v tomto vývojovém prostředí a o novinkách verze 4 se zde zmíním jen okrajově v následujícím seznamu:

- V prvé řadě bych vyzdvihl WYSIWYG editor, který už byl velice kvalitní (snad nejspolehlivější WYSIWYG editor, se kterým jsem se setkal) v předchozí verzi. Nyní však musí pracovat se dvěma komponentovými frameworky, což zvládá dobře.

- Jelikož podstatnou část návrhu aplikace stráví programátor psaním zdrojového kódu, je jakékoliv intuitivní napovídání editorem neocenitelným pomocníkem. Ve verzi 4 bylo toto napovídání vylepšeno oproti verzi 3 tak, že inteligentně filtruje naši nabídku podle kontextu. Dále v tooltipu se nám zobrazují kontextové nápovědy. Také zde byla opravena navigace v textu formou „ctrl + klik myší na funkci“. Mnohokrát jsem se setkal s událostí, kdy mi tato funkce ve verzi 3 vůbec nefungovala.

- Co stojí za zmínku a podle mého názoru patří mezi jednu z největších novinek je, že se věnovala velká pozornost wizardům pro napojení na datové zdroje. Vyřešilo se tím napojování na různé typy datových zdrojů a je se možné napojit na libovolný, takzvaný backend. Ukázku takového napojení Flex 4 a php je možno vidět v tomto článku [7].

- Další velmi užitečná novinka je integrace takzvaného Network Monitoru, který je zabudovaný přímo v rozhraní Flex Builderu 4, a který je určen primárně pro ladění a sledování požadavků a přenosu dat mezi klientem a serverem.

- Poslední zmíněnou novinkou je vylepšení debugeru, kde je mnohem lépe zpracováno okno expressions. Potom je zde velice užitečné vylepšení, které určitě potěší, a tím je možnost vložení podmíněných brakpointů, které mi osobně při práci dost chyběly, a jejich náhradu jsem musel řešit jinou cestou.

7 Rezervace sportovišť praktická aplikace teoretických poznatků

V této bakalářské práci je obsažena jak část teoretická, tak část praktická a tudíž se v této kapitole budu zabývat návrhem vlastní aplikace a systémem rezervací sportovišť v areálu VŠB - TUO v Porubě.

7.1 Analýza informačního systému rezervace sportovišť

Předtím, než jsem začal se samotným programováním aplikace v Adobe Flex Builderu, bylo nutné vytvořit analýzu tohoto systému. V prvé řadě jsem si vytvořil konceptuální model mé aplikace, abych věděl, jak bude má datová struktura vypadat a jak budou tabulky mezi sebou propojeny. Poté následovalo vytvoření Use Case Diagramu a Data Flow Diagramu k určení všech potřebných procesů a datových toků. Na závěr jsem ještě vytvořil Sekvenční Diagramy, které mi zobrazily detailnější chování složitějších procesů. V této práci byly pro vytvoření sekvenčních diagramů vybrány jen ty nejdůležitější a nejsložitější procesy.

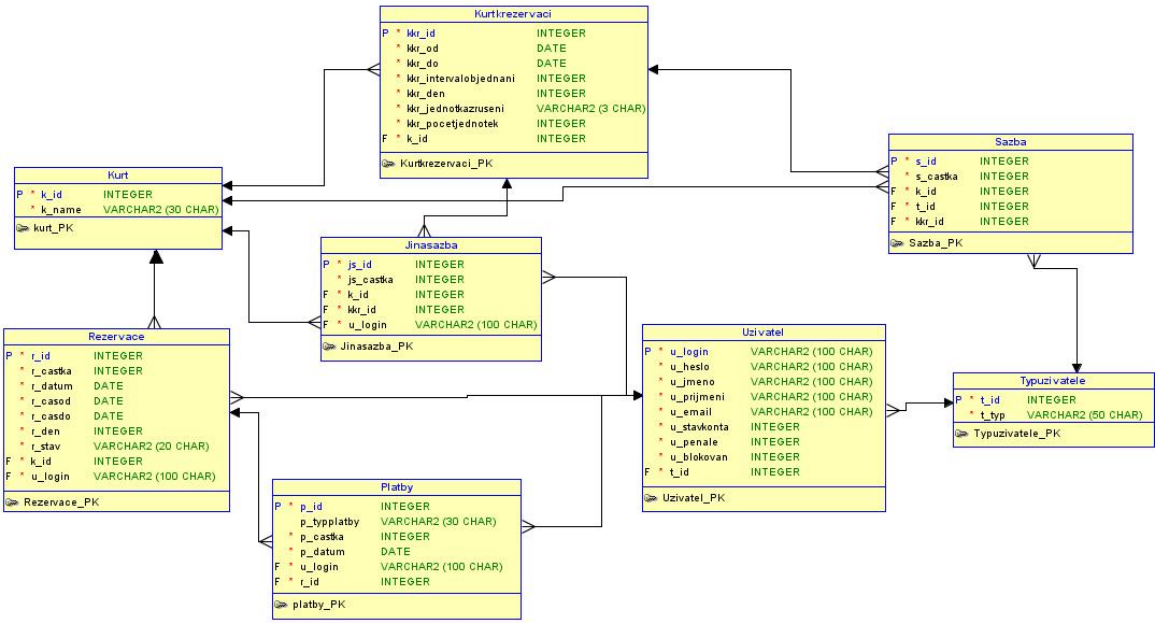
7.1.1 Proč

Úkolem je vytvořit rezervační online systém za pomocí zvolené technologie RIA, ve kterém by mělo být možné provádět všechny základní operace, které se vyskytují v ostatních rezervačních systémech na internetu. Dále tento systém obohatit o prvky RIA a vyzdvihnout tímto můj rezervační systém a ukázat na potenciál těchto technologií.

7.1.2 Konceptuální model

Prvním krokem po vybrání vhodné technologie a vývojového prostředí pro tvorbu mé práce bylo vytvoření optimální datové struktury, která zahrne celou problematiku rezervace sportovišť. Databáze musela být navrhnuta tak, aby splnila všechny požadavky, jak uživatele, tak administrátora systému. Výsledná datová struktura obsahuje 8 tabulek, které bylo potřeba k realizování rezervačního systému a jejichž výslednou podobu můžete vidět na následujícím ER diagramu.

V tabulce *Kurt* jsou uchovány názvy všech našich sportovišť, které v systému využívám.V tabulce *Kurtkrezervaci* se zase skladují data, díky kterým nastavím otevírací dobu jednotlivých sportovišť a dobu jednoho rezervačního intervalu, který si můžou uživatelé objednat. Například lze nastavit, že čas v pondělí od 8:00 do 12:00 se rozdělí na jednotlivé rezervace v intervalu 40 minut, takže nám vznikne 6 možných rezervací v době mezi osmou a dvanáctou hodinou dopolední, na které je možné se přihlásit. Ke každému takovému času pak administrátor přidělí čas, do kterého nejpozději před začátkem dané rezervace má možnost uživatel tuto rezervaci zrušit.



Obrázek 10: Konceptuální model - ER diagram

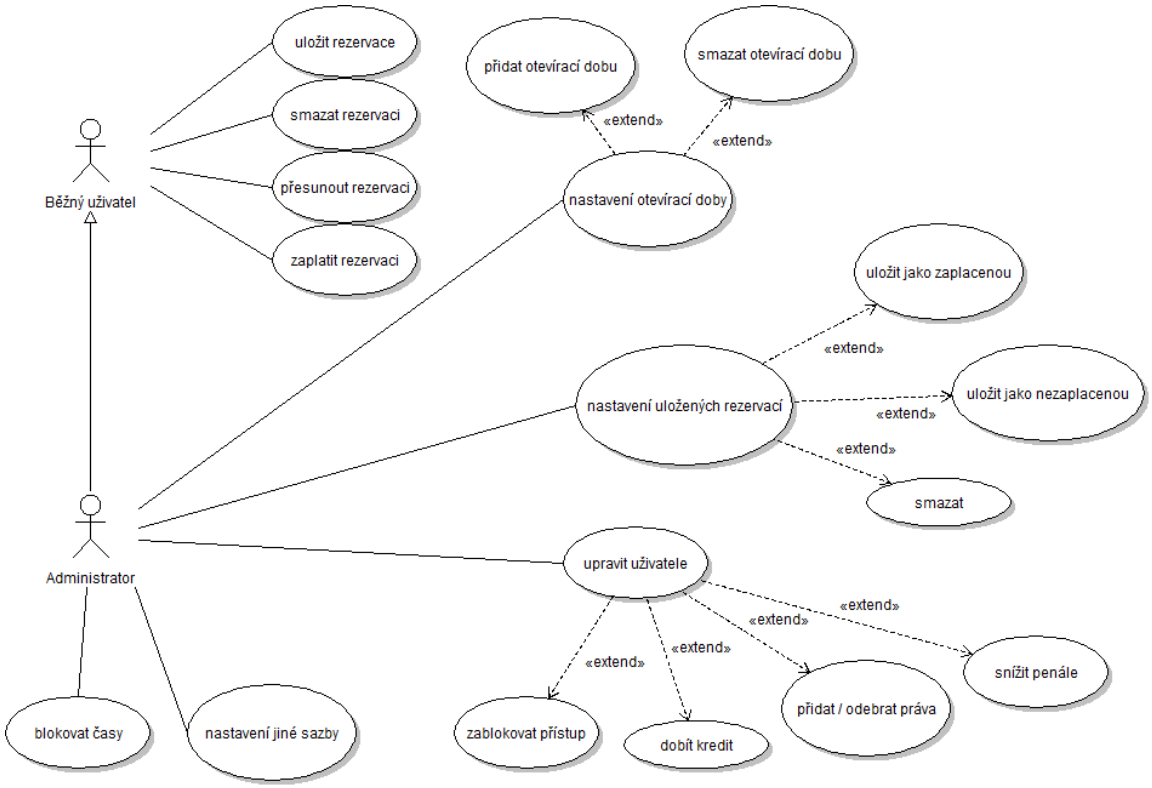
Dále bylo třeba vyřešit cenu jednotlivých rezervací. Pro tento účel byla vytvořena tabulka *Sazba* a tabulka *Jinasazba*. Do první tabulky *Sazba* se nám ukládají ceny přiřazené jednotlivým časům, které jsme vytvořili. Tyto ceny platí pro každého uživatele rezervačního systému, který nemá vytvořený žádný záznam v tabulce *Jinasazba*. Pokud se rozhodneme zadat některému uživateli jiné ceny rezervací, uloží se tyto částky do tabulky *Jinasazba*. Systém už pak pracuje pouze s touto tabulkou u tohoto daného uživatele. Tím můžeme zajistit jedinečnou cenu kterémukoliv uživateli. Tabulka *Platby* slouží k uchovávání veškerých plateb provedených buď při placení rezervace nebo při zakoupení kreditu pro platby přímo z mého rezervačního systému. V mé datové struktuře nesmí chybět tabulka *Uzivatel*, ve které se ukládají veškerá data o uživateli od přihlašovacího jména a hesla až po stav kreditu či penále, které musí uživatel vyrovnat. S touto tabulkou souvisí další a to *Typuzivatele*, ve které se ukládá, jestli se jedná o prostého uživatele nebo o administrátora, který může spravovat jednotlivá sportoviště. Poslední tabulka, která nám zbývá je *Rezervace*, kde jsou uloženy veškeré rezervované časy a stav dané rezervace, jestli je zaplacená či nikoliv.

7.1.3 Use case diagram

V další části bylo třeba vytvořit jednoduchý use case diagram, abych měl lepší přehled o tom, jaké akce se budou v mém systému provádět.

V prvé řadě jsem si stanovil, kdo bude můj systém využívat. V mém případě to jsou dva různé typy uživatelů. První je obyčejný uživatel, který nebude mít přístup do administrace a který si bude moci pouze rezervovat a spravovat své rezervace.

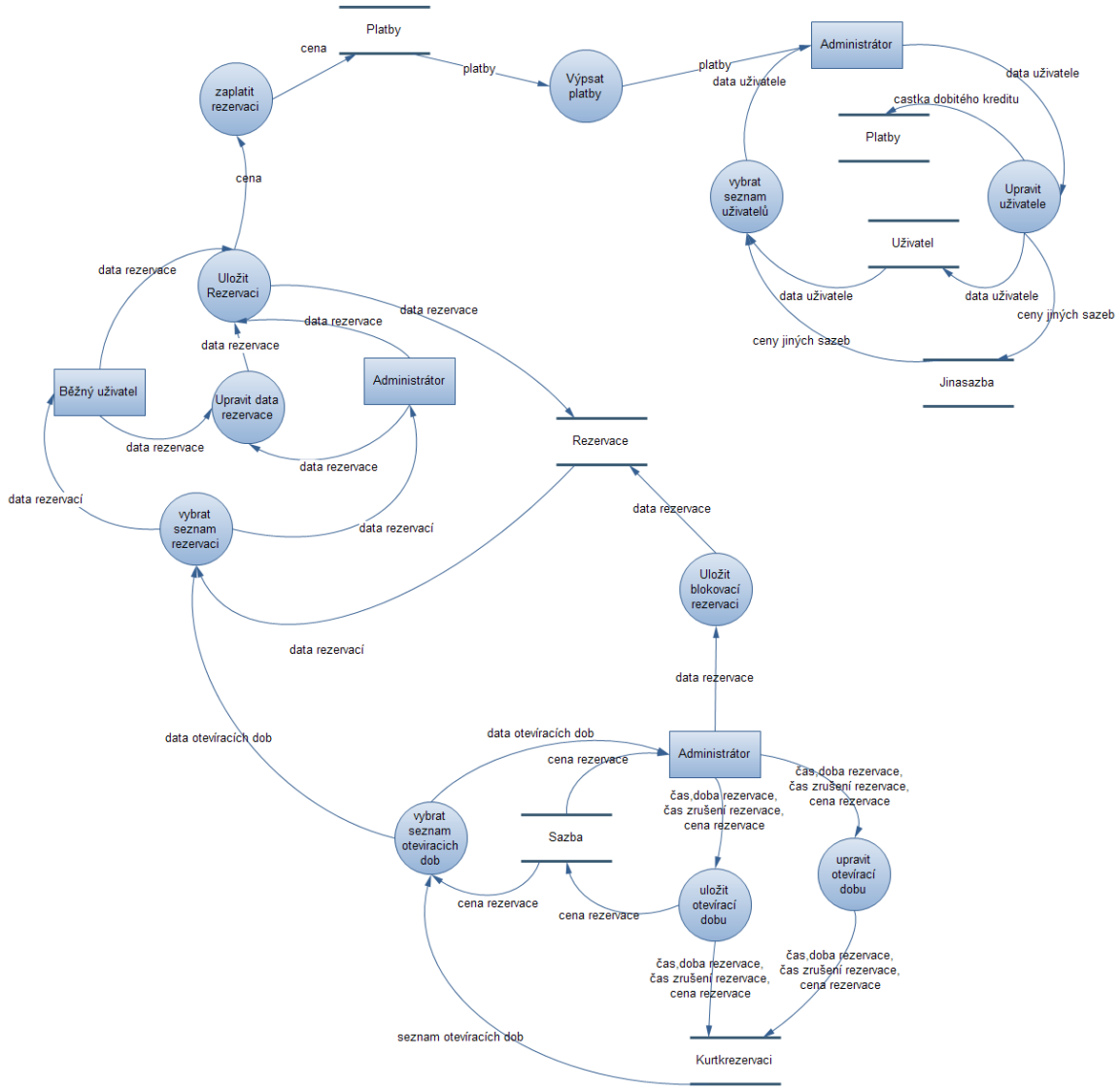
Druhým typem uživatele je Administrátor, který bude mít stejné možnosti jako běžný uživatel a navíc mu budou umožněny operace, které administrují jednotlivá sportoviště. Dále bude moci provádět akce, které upravují data běžným uživatelům. Mezi tyto akce patří například nastavení jiných sazeb uživatelům, dobití kreditu, přidání respektive odebrání administrátorských práv, smazání penalizace, či zablokování přístupu do systému z důvodu velkého penále nebo jiného přestupku. Administrátor dále bude moci upravovat rezervace uložené uživateli a to tak, že danou rezervaci smaže nebo uloží jako zaplacenou respektive nezaplacenou.



Obrázek 11: Use Case Diagram

7.1.4 Data Flow Diagram

Dalším krokem k analyzování mého rezervačního systému bylo popsat funkčnost mého systému a vytvořit jeho grafickou prezentaci datových toků, takzvaný DFD (Data Flow Diagram), abych si mohl znázornit kudy vedou datové toky a jakým procesem je daný tok zpracováván, kdo používá daný proces a následně, kde jsou zpracovaná data uložena. Jednotlivé procesy, které využívám a které jsou znázorněny v mém DFD, vycházejí z mého předchozího Use Case Diagramu.



Obrázek 12: Data Flow Diagram

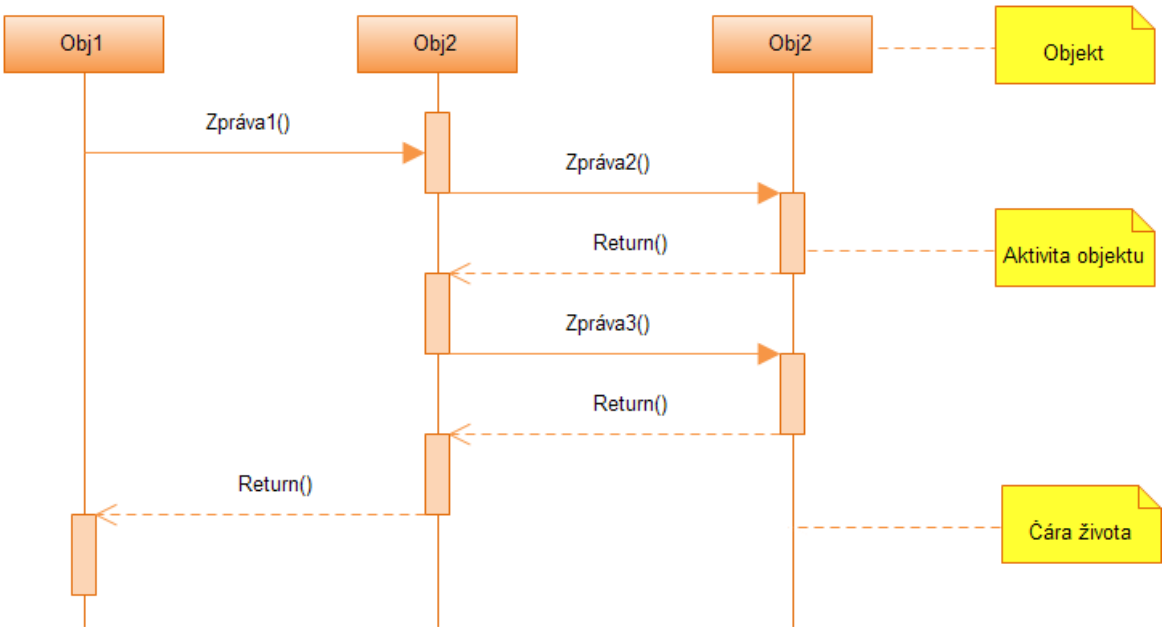
Nejprve jsem si určil takzvané terminátory, které představují externí entity, které komunikují se systémem a stojí vně systému. Jelikož můj rezervační systém neobsahuje žádné organizace, oddělení ani úřady, které by se systémem nějak pracovaly, zbyly mi terminátory určené jen jako osoby. To jsou *běžný uživatel* a *administrátor*, tak jak je vykresleno v předchozím Use Case Diagramu. Dále jsem si musel vytvořit procesy, které jsem v našem systému používal. Některé procesy jsou zobecněny, jelikož vykreslit detailní chování by bylo zbytečné. Takže namísto vykreslování zvlášť procesů smazání, přesunutí, zaplacení nebo nezaplacení rezervace jsem tyto sjednotil do jednoho, a to *upravit data rezervace*. Na závěr jsou v DFD použita takzvaná *data store* neboli *úložiště dat*. Jelikož k tabulkám *typuzivatele* a *kurt* se

nevztahují žádné procesy, nejsou v mém DFD použity, a tím mi v mém diagramu zbylo šest datových úložišť.

7.1.5 Sekvenční Diagram

Posledním krokem k dokončení mé analýzy je vytvoření Sekvenčních Diagramů. Tyto diagramy mi detailněji ukázaly, jak se budou chovat jednotlivé procesy, a protože by těchto diagramů bylo mnoho, použil jsem jen ty nejdůležitější.

Sekvenční Diagram v podstatě zachycuje sekvenci interakce mezi objekty tříd, ke které dochází při komunikaci (předávání činnosti) v systému. V Sekvenčním Diagramu jsou znázorněny *objekty* a jejich takzvaná *lifeline* neboli *čára života*. Tato *čára života* ukazuje do jaké doby objekt žije a nezanikne. V mém případě jsou všechny tyto čáry stejně dlouhé.

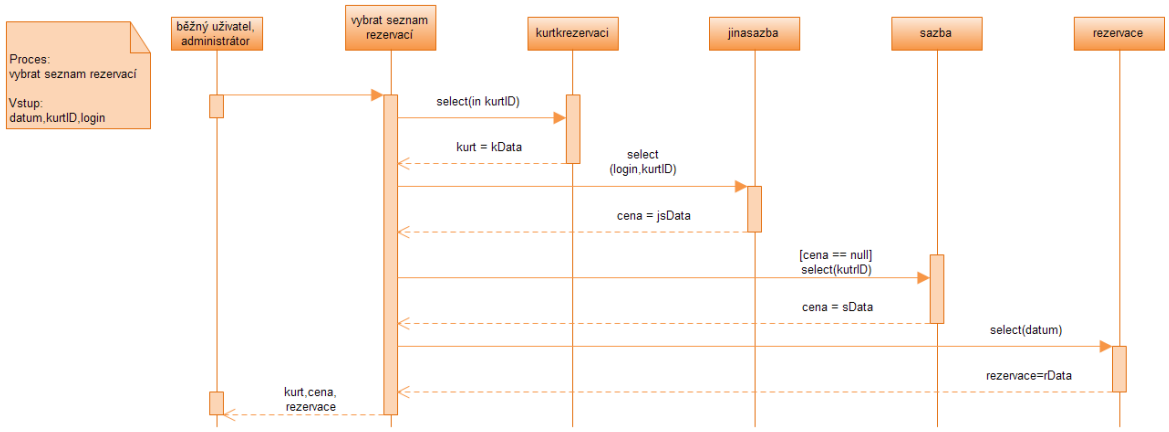


Obrázek 13: Ukázka sekvenčního diagramu

Pak je zde takzvaná *aktivita objektu* a ta znázorňuje, kdy je daný objekt aktivní. Dále jsou zde zprávy, které jsou zaslány mezi objekty a jsou znázorněny plnou šipkou. Od těchto zpráv je očekávaná návratová hodnota. Dále jsou zde zprávy s prostou šipkou takzvané *asynchronní*. Po zaslání této zprávy klient nečeká na návratovou hodnotu a pokračuje dále s příkazy. Poslední věc, která se v mém Sekvenčním Diagramu vyskytuje, jsou popisky pro upřesnění obsahu.

Prvním Sekvenčním Diagramem je vybrání seznamů rezervací. Z tabulky *kurtkrezervaci* vyberu požadované otevírací doby. K nim potřebuji vybrat částku rezervace pro daného uživatele z tabulky *jinasazba*. Pokud v této tabulce nejsou k danému uživateli žádná data, vyberu z tabulky *sazba* částku, která je pro všechny uživatele, kteří nemají svou specifickou sazbu v tabulce

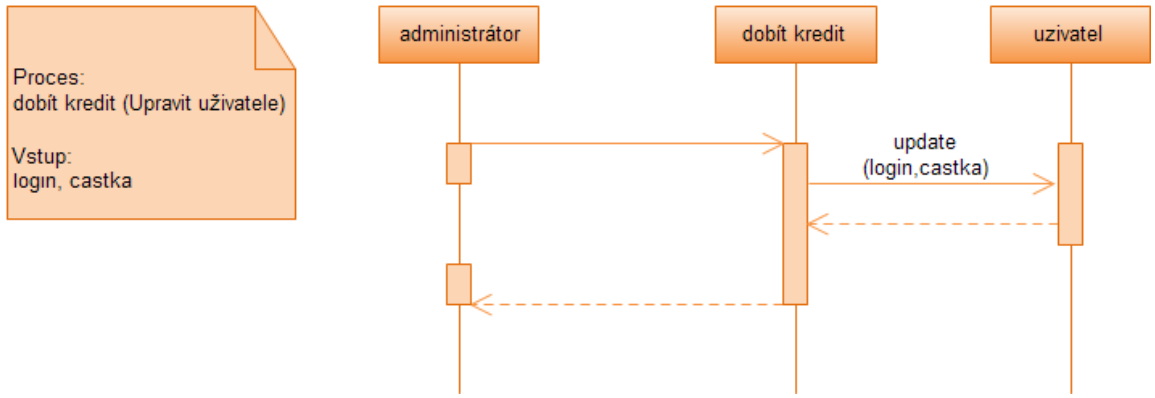
jinasazba. Nakonec vyberu data z tabulky *rezervace*, abych mohl získat již uložené rezervace a zjistit tak, která data jsou volná a která ne.



Obrázek 14: Sekvenční diagram - seznam rezervaci

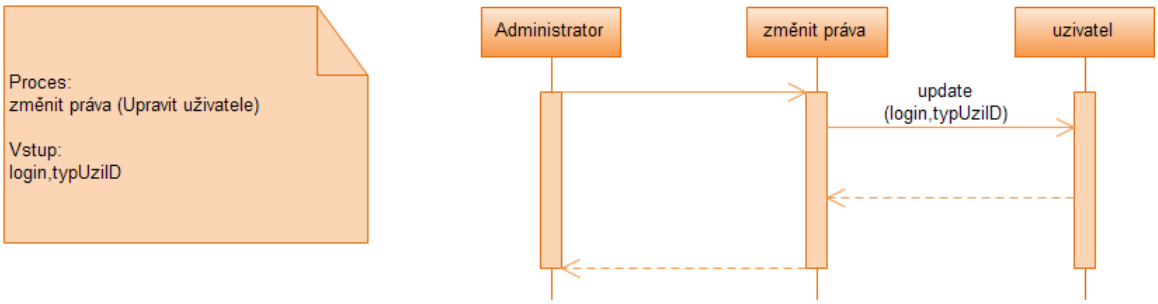
Proces upravit uživatele znázorněný v mém DFD se dělí na další podprocesy. Jelikož tyto podprocesy nejsou složité a jsou v podstatě skoro stejné a jejich jednotlivé znázornění v DFD by zbytečně zabralo spoustu místa a mohlo by vést ke zmatení, spojil jsem tyto do jednoho obecnějšího procesu. Z toho důvodu znázorním alespoň dva tyto podprocesy v sekvenčním diagramu, aby měl čtenář možnost si představit jak tyto podprocesy vypadají. Procesy nejsou složité, tudíž nepotřebují detailnější vysvětlení.

První z těchto dvou procesů znázorňuje dobití kreditu uživateli administrátorem.



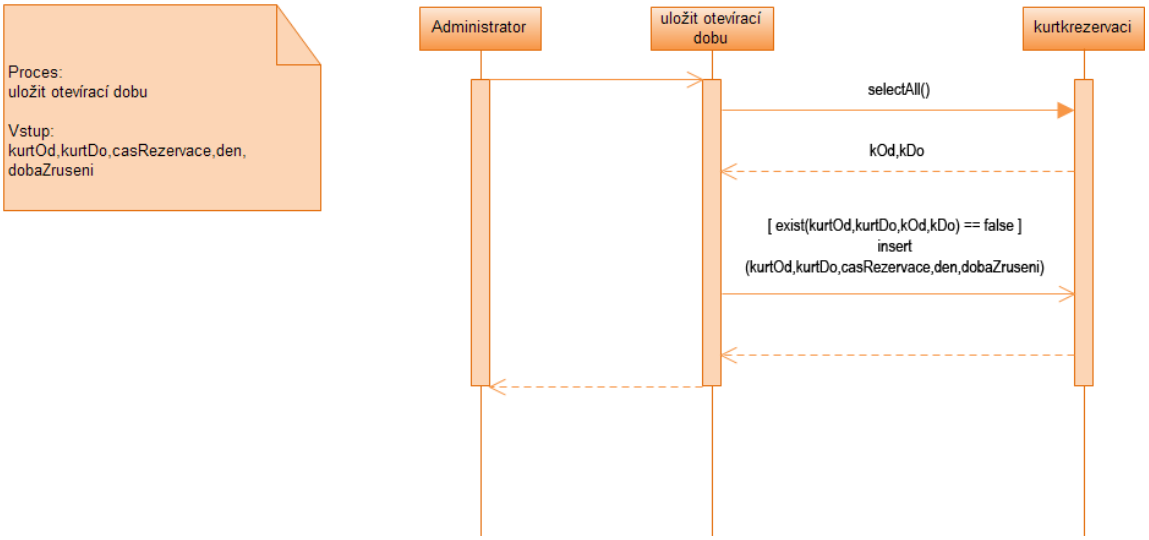
Obrázek 15: Sekvenční diagram - dobití kreditu

Druhý proces zobrazuje změnu práv uživateli administrátorem.



Obrázek 16: Sekvenční diagram - změna práv

Posledním sekvenčním diagramem, o kterém se v mé práci zmiňuji, je uložení otevírací doby sportoviště. Hned na začátku je třeba vybrat všechny otevírací doby, abych zjistil, jestli už se vkládaná doba v databázi nevyskytuje. Pokud tomu tak není, rezervace může být vložena.



Obrázek 17: Sekvenční diagram - uložit otevírací dobu

7.2 Implementace informačního systému rezervace sportovišť

Po dokončení celkové analýzy informačního systému bylo možné začít pracovat na samotné implementaci mé aplikace. V této části kapitoly shrnu kroky mé implementace, jak jsem postupoval, co vše se do mé aplikace muselo zakomponovat, na co bylo třeba se zaměřit a jaké případné problémy se při programování vyskytly.

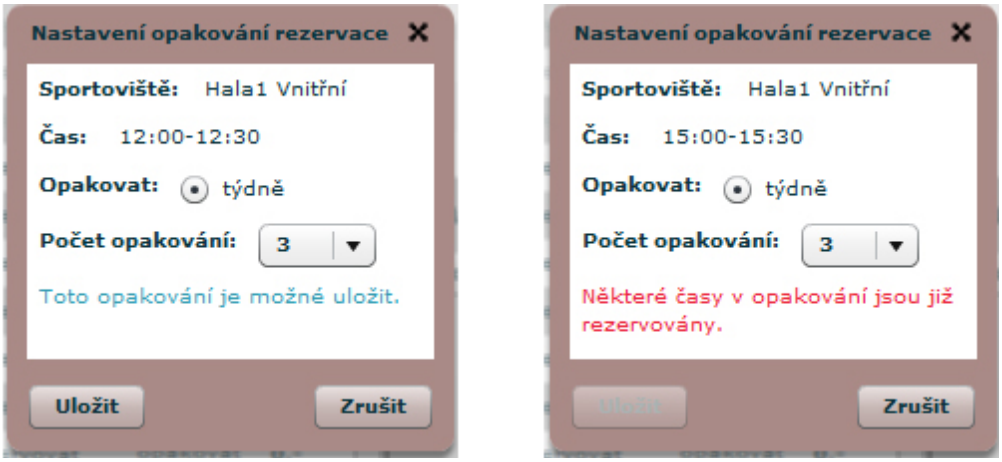
Jako první věc bylo nutné navrhnout uživatelsky přívětivý design stránek. Zaměřil jsem se na to, aby grafika nebyla zbytečně matoucí, či aby nějak jinak neomezovala uživatele v prohlížení těchto stránek. Grafický layout musel rovněž mít rysy sportovního charakteru. Jelikož se jedná o rezervace sportovišť různých typů sportovních aktivit, rozhodl jsem se zvolit tématiku míčů z různých odvětví sportu.

Obrázek 18:Hlavička stránky

Výběr sportovišť jsem rovněž řešil grafickou formou z důvodu přehlednosti areálů a jejich jednotlivých sportovišť (viz Obrázek 6 a Obrázek 7). Uživatel tak přesně ví, na které sportovní hřiště si rezervaci zadává.

Dalším krokem bylo vytvoření systému rezervací a jejich zobrazení. Zvolil jsem takový systém, který pracuje s intervaly otevíracích dob, které se dělí na jednotlivé rezervace o určité délce určené administrátorem. Administrátor si tak jen vytvoří otevírací časy daného sportoviště, délku rezervace v tomto intervalu a cenu rezervace. O vytvoření jednotlivých rezervací se pak postará už samotný systém. Pro práci se všemi rezervacemi, ať už se jedná o jejich mazání, úpravy, blokace, či jiné editace, jsem vytvořil třídu **Rezervace**. Ta se stará o všechny tyto akce, které jsou prováděny s vytvořenými rezervacemi. Při vytváření funkcí pro ukládání jednotlivých rezervací nebyl žádný problém, když se jednalo o rezervace bez opakování. Jakmile jsem začal řešit opakování rezervací, bylo třeba ošetřit několik věcí, aby nedošlo k uložení více rezervací různými uživateli na stejný čas a datum. Bylo tedy nutné upozornit uživatele, zda dané opakování rezervace může uložit či ne. Pokud se v námi určeném opakování vyskytla již uložená nějaká cizí rezervace, bylo třeba zamezit v uložení takového opakování (viz Obrázek 19).

Dále bylo třeba vytvoření sekce, kde si uživatel své rezervace může přesunout, smazat nebo zaplatit nezaplacené rezervace a sekce, kde si bude moci změnit své heslo. Naprogramovat smazání a zaplacení rezervace nebylo nijak zvlášť obtížné. Složitější bylo, když jsem se pustil do implementace přesunutí rezervace. Nabídka přesunutí rezervace se otevírá v novém takzvaném „popup“ okně a obsahuje veškeré potřebné údaje o vybrané rezervaci. Poté jsem musel naprogramovat generování volných časů rezervací na dané datum. Ty se zobrazují v tomto okně a z nich si uživatel vybírá, na který čas danou rezervaci přesune. Je zde rovněž kalendář pro vybrání data, na který se rezervace chce přesunout, a podle kterého se výpis volných časů aktualizuje.



Obrázek 19: Opakování rezervace - povolení a zamezení

Asi nejsložitější a nejdůležitější z celého programování bylo vytvoření administrační sekce stránek. Uživateli přihlášenému jako Administrátor, bylo třeba umožnit všechny potřebné úkony, které jsou třeba pro úpravu sportovišť či běžných uživatelů. Celou rezervační sekci jsem rozdělil do sedmi záložek, ve kterých se administrují buď data týkající se vybraného sportoviště nebo uživatelé z celého systému.

- **Nastavení otevírací doby** – Základem zde bylo vkládání otevíracích časů daného sportovního hřiště a mazání těchto časů. Při mazání jsem musel ošetřit stav, kdy se v dané otevírací době vyskytuje již zabukovaná něčí rezervace. Pokud se takový stav vyskytne, je tato rezervace smazána a uživateli jsou vráceny peníze za danou rezervaci zpět v případě, že tato rezervace ještě nebyla uskutečněna a zároveň již byla zaplacena. Při vkládání nového času jsem zase musel vytvořit takovou funkci, která mi zjistí, jestli se zadávaný čas otevírací doby nekryje s nějakou již vytvořenou otevírací dobou v daném dni.

- **Výpis rezervací** – Tato další sekce v administraci je vytvořena za účelem úpravy rezervací, které byly vloženy uživateli do systému. V této sekci je možné takovou rezervaci uložit jako zaplacenou, nezaplacenou nebo ji smazat. V případě smazání již zaplacené rezervace jsou peníze za tuto rezervaci vráceny uživateli na účet.

- **Nastavení jiné sazby uživateli** – Důležitá část administrace, kde bylo třeba vytvořit systém pro nastavování jiných sazeb za rezervace zvolenému uživateli. Při nastavování otevíracích časů sportovišť v první sekci administrace se zadává cena. Tato cena je jednotná pro všechny, kromě těch, kteří mají nastavenou jinou specifickou sazbu vytvořenou v této sekci. Pokud má někdo vytvořenou vlastní sazbu, systém nebere v úvahu jednotnou sazbu pro všechny, ale načte si z tabulky *Jinasazba* ceny, které jsou učeny tomuto uživateli.

- **Upravit uživatele** – V této sekci je možné upravit registrované uživatele. Jsou zde funkce jako je dobití kreditu, přiřazení či odebrání administrátorských práv nebo zablokování uživatele, aby již nadále neměl přístup do systému rezervace sportovišť.

- **Blokovat časy** – Tato možnost v administraci nesměla chybět. Administrátorovi by mělo být umožněno blokovat jednotlivé časy, protože například o státním svátku bude jistě sportoviště zavřeno, či jinak časově omezeno. V takovém případě si administrátor sportoviště zablokuje určité časy, kdy je sportoviště zavřeno. Opět, když je daný čas již někým rezervován, jsou peníze v případě zaplacení této rezervace vráceny zpět původnímu majiteli.

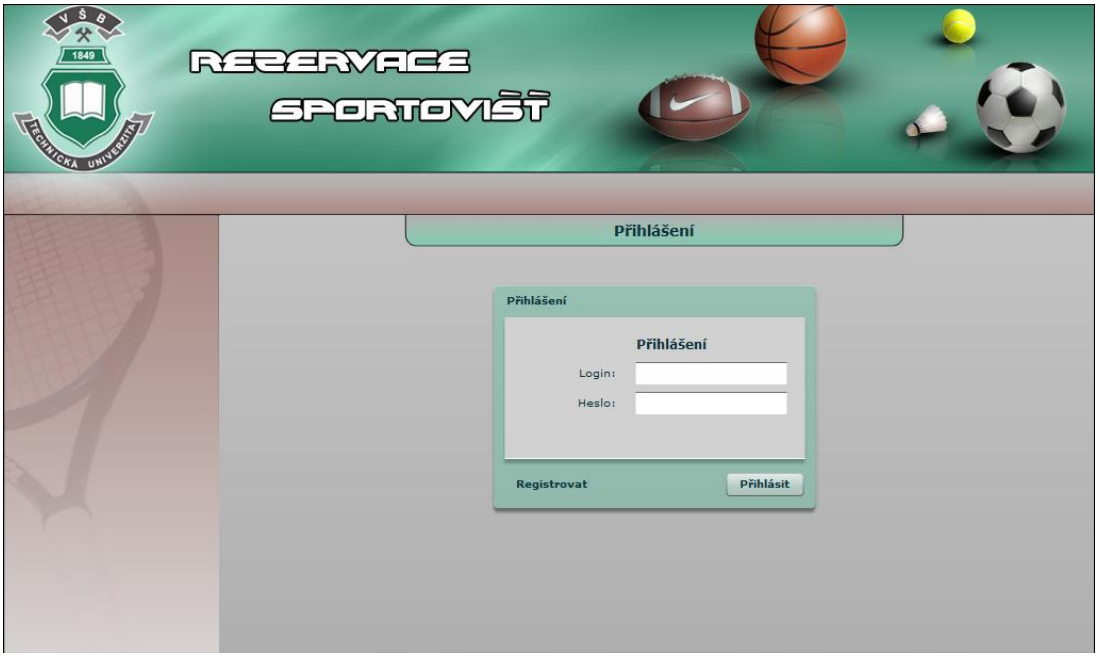
- **Dlužníci** – V sekci dlužníci je vytvořen seznam dlužníků. Můžeme zde zobrazit dlužníky pouze z daného sportoviště nebo zobrazit dlužníky ze všech sportovišť. Takovéto uživatele je možno v této sekci buď zablokovat nebo pokud na recepci dluh zaplatí, tak dluh z databáze vymazat. Po zaplacení dlužné částky na recepci má administrátor možnost smazat uživateli dluh buď z daného sportoviště, nebo smazat dluh ze všech sportovišť. Čili pokud uživatel dluží na jednom sportovišti 30 Kč a na druhém 40 Kč, může být smazán dluh z jednotlivých sportovišť zvlášť nebo může být smazán kompletní dluh 70 Kč.

- **Seznam plateb** – Poslední sekce, kterou bylo třeba naimplementovat, byl výpis všech plateb, které byly uskutečněny v jednotlivých měsících, aby měl administrátor přehled o provedených platbách.

7.2.1 Vizuální ukázky aplikace

V této kapitole jsou znázorněny vizuální ukázky mé aplikace. Čtenář, který nebude mít přístup k aplikaci tak bude mít představu, jak tato aplikace vypadá.

Při spuštění aplikace na webovém prohlížeči se uživateli zobrazí úvodní stránka s přihlašovacím formulářem. Zde se může uživatel rovněž přepnout na formulář pro registraci.

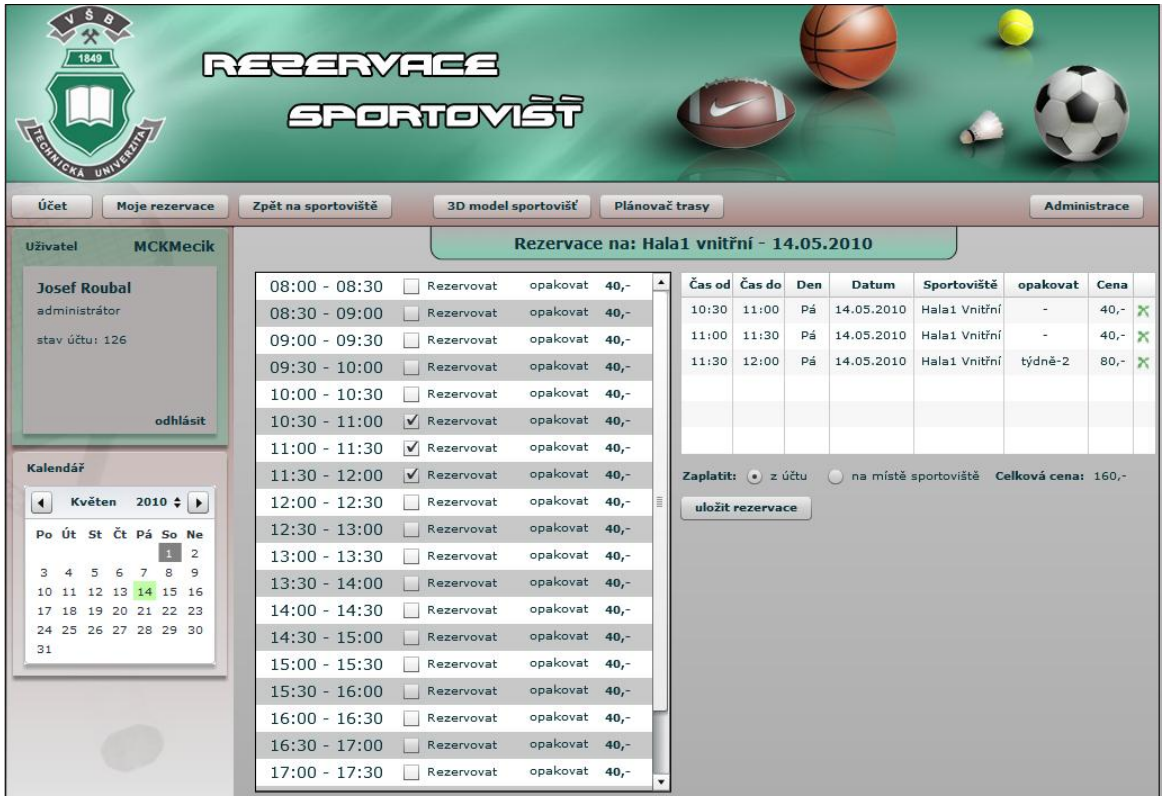


Obrázek 20: Ukázka aplikace – přihlášení

Po přihlášení se uživatel dostane na stránku, kde v levém sloupci vidí své údaje a stav svého účtu. Na horní liště je navigační menu a v hlavní části obrazovky je znázorněn grafický výběr sportovišť.

Obrázek 21: Ukázka aplikace - výběr sportovišť

Po vybrání sportoviště je uživateli zobrazena stránka s výběrem jednotlivých časů rezervací. V levém sloupci je kalendář pro výběr data. Při změně data jsou časy aktualizovány na určené datum.

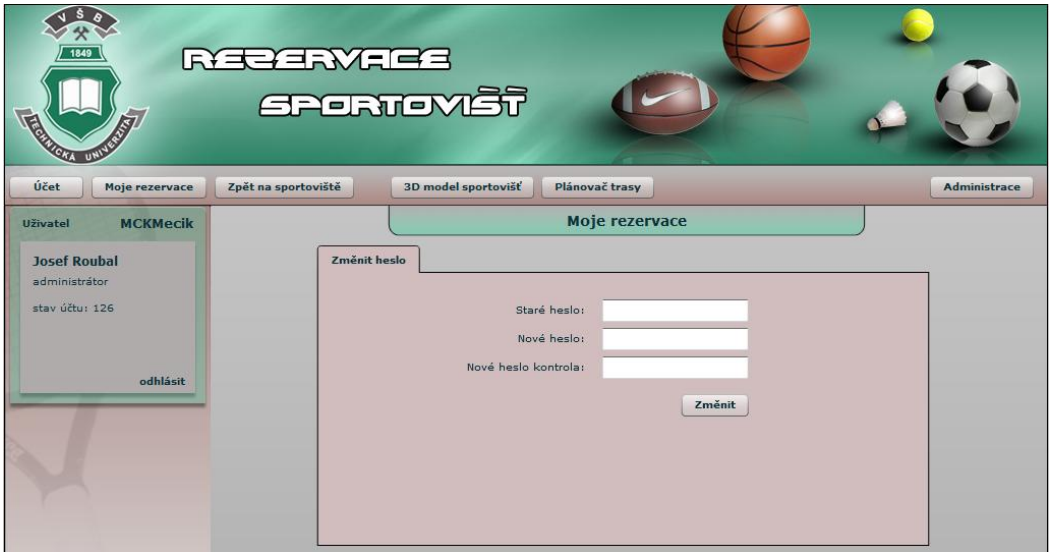


Obrázek 22: Ukázka aplikace - výběr časů

Na následujícím obrázku je znázorněna ukázka administrační sekce, kde si uživatel vybírá areál z první řady tlačítek a poté konkrétní sportoviště z druhé řady tlačítek. Následně se mu objeví administrační menu pro dané sportoviště, kde jsou jednotlivé záložky pro administraci.

Obrázek 23: Ukázka aplikace – administrace

Po stisknutí tlačítka **Účet** se uživatel dostane do sekce, kde si může změnit své heslo.



Obrázek 24: Ukázka aplikace - změna hesla

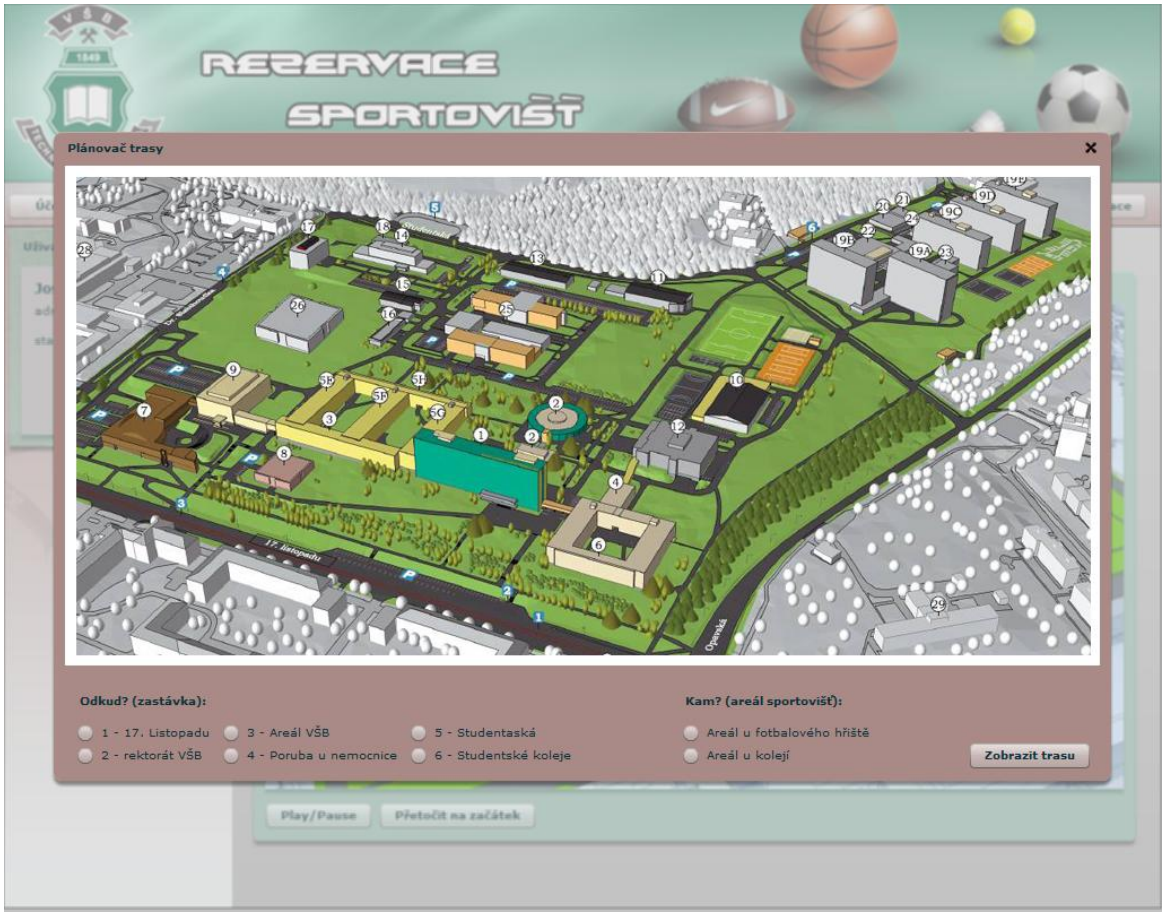
V sekci **Moje rezervace** se uživateli zobrazí možnosti **Smazat rezervaci**, **Přesunout rezervaci**, **Zaplatit rezervaci**. Po vybrání jedné z možností se zobrazí vždy seznam nadcházejících rezervací.

Obrázek 25: Ukázka aplikace – Moje rezervace

Aby si uživatel mohl prohlédnout 3D model areálu VŠB – TUO, stiskne tlačítko **3D model sportovišť**. Poté se mu zobrazí 3D model areálu, který je možno pomocí tlačítka **Play/Pause** zastavit, či pustit a nebo přetočit zpět na začátek pomocí tlačítka **Přetočit na začátek**.

Obrázek 26: Ukázka aplikace - 3D model

Jako poslední ukázka je plánovač trasy, kde si uživatel vybere ze seznamu zastávek v sekci **Odkud** počátek trasy a v sekci **Kam** konec trasy a po stisku tlačítka **Zobrazit trasu** se vykreslí nejbližší cesta od zvolené zastávky k danému areálu. Při vykreslování trasy se rovněž zobrazí tlačítka **Přehrát od začátku** pro zopakování trasy a **Nový výběr trasy** pro zvolení nového počátečního a koncového bodu.



Obrázek 27: Ukázka aplikace - Plánovač trasy

7.2.2 Přístupové údaje pro testování praktické části

Praktická část je uložena na stránkách: www.mecik.ic.cz
Pro otestování kompletní funkčnosti praktické části je nutné se přihlásit jako administrátor:

Login: admin
Heslo: admin

8 Závěr

Úkolem této bakalářské práce bylo získat přehled v oblasti RIA technologií a seznámit čtenáře s jednotlivými nástroji, které jsou dostupné pro tvorbu RIA. Zároveň čtenáři ukázat, jaké jsou výhody či nevýhody jednotlivých technologií a poukázat na jejich potenciál oproti klasickým internetovým stránkám. Dalším úkolem bylo vytvořit praktickou ukázku využívající tyto technologie. Pro tuto ukázku jsem zvolil rezervační systém sportovišť, vytvořen v programu Adobe Flex Builder 3. Jelikož jsem vybral právě tento typ rezervačního systému, bylo nutné čtenáře seznámit s jeho problematikou, včetně všech finančních aspektů. Tento systém si je možné prohlédnout na přiloženém médiu, kde se nachází i manuál pro jeho použití a programátorská dokumentace.

Při tvorbě aplikace, jsem kladl důraz na to, aby byly stránky intuitivní a jednoduché. V rámci RIA technologií, objevující se v mé praktické části, jsou to převážně 3D model, efekty při přechodech mezi různými stavy aplikace nebo zobrazování dat bez nutnosti vykreslování celé stránky. Přínos zmíněných technologií, a nejen těchto, je více než jasný už jen ze samotného názvu Rich Internet Aplication. Uživatel má větší potěšení z užívání takových propracovanějších stránek. Ten, kdo se rozhodne investovat do těchto technologií, jistě přiláká spoustu uživatelů, protože podle mého názoru v dnešní době nejde jen o jednoduchost a funkčnost, ale také o vizuální a animační prožitek, který tyto technologie zaručeně umožňují.

9 Reference

[1] Cristian Darie, Bogdan Brinzarea, Filip Chereches Tosa, Mihai Bucica, *AJAX a PHP tvoříme interaktivní webové aplikace PROFESIONÁLNĚ.* Brno : ZONER software, s.r.o., 2006. 80-86815-47-1

[2] III., Anthony T. Holdener, *Ajax:The Definitive Guide.* USA : O'Reilly Media,Inc., 2008. 978-0-596-52838-6

[3] Silverlight 3, http://www.lupa.cz/clanky/silverlight-3-ie-8-a-dalsi-novinky-microsoftu/

[4] John Papa, *Silverlight - datové služby.* Brno : Zoner Press, 2009. 978-80-7413-041-0

[5] Matthew MacDonald, *Pro Silverlight 3 in C#.* USA : Springer-Verlag New York, inc., 2009. 978-1-4302-2382-5

[6] Joshua Noble, Todd Anderson, *Flex 3 CookBook.* USA : O'reilly Media, Inc., 2008. 978-0-596-52985-7

[7] Spojení Flex 4 a PHP ve Flash Builderu 4, http://zdrojak.root.cz/clanky/spojeni-flex-4-a-php-ve-flash-builderu-4/

[8] Budujte webové prezentace nové generace, http://www.amsoft.cz/Produkty/Adobe/flex/overview.html

[9] ADOBE FLEX BUILDER 3, http://www.amsoft.cz/Produkty/Adobe/flex/builder.html

[10] Adobe AIR, http://www.adobe.com/cz/products/air/

[11] Rich Internet Applications dneška, http://blackmartin.yweb.cz/2008/06/12/rich-internet-applications-dneska/

[12] AJAX, http://cs.wikipedia.org/wiki/AJAX

[13] AJAX – kde jsou hranice, http://www.snizekweb.cz/clanky/ajax-kde-jsou-hranice/

[14] Silverlight - co je Silverlight?, http://interval.cz/clanky/silverlight-co-je-silverlight/

Seznam příloh